



KUKA System Technology
KUKA.Ethernet KRL 3.1
For KUKA System Software 8.6



Issued: 21.09.2018
KST Ethernet KRL 3.1 V1
KUKA Deutschland GmbH

© Copyright 2018

KUKA Deutschland GmbH

Zugspitzstraße 140

D-86165 Augsburg

Germany

This documentation or excerpts therefrom may not be reproduced or disclosed to third parties without the express permission of KUKA Deutschland GmbH.

Other functions not described in this documentation may be operable in the controller. The user has no claims to these functions, however, in the case of a replacement or service work.

We have checked the content of this documentation for conformity with the hardware and software described. Nevertheless, discrepancies cannot be precluded, for which reason we are not able to guarantee total conformity. The information in this documentation is checked on a regular basis, however, and necessary corrections will be incorporated in the subsequent edition.

Subject to technical alterations without an effect on the function.

KIM-PS5-DOC

Translation of the original documentation

Publication: Pub KST Ethernet KRL 3.1 (PDF) en
PB11613

Book structure: KST Ethernet KRL 3.1 V1.1
BS10664

Version: KST Ethernet KRL 3.1 V1

Contents

1	Introduction.....	6
1.1	Target group.....	6
1.2	Industrial robot documentation.....	6
1.3	Representation of warnings and notes.....	6
1.4	Terms used.....	7
1.5	Trademarks.....	8
1.6	Licenses.....	8
2	Product description.....	9
2.1	Overview.....	9
2.2	Configuration of an Ethernet connection.....	9
2.2.1	Behavior in the event of a lost connection.....	9
2.2.2	Monitoring a connection.....	10
2.3	Data exchange.....	10
2.4	Saving data.....	11
2.5	Client-server mode.....	12
2.6	Protocol types.....	12
2.7	Event messages.....	13
2.8	Error treatment.....	13
2.9	Intended use.....	13
3	Safety.....	15
4	Installation.....	16
4.1	System requirements.....	16
4.2	Installation via WorkVisual.....	16
4.2.1	Installing or updating KUKA.Ethernet KRL 3.1.....	16
4.2.2	Uninstalling KUKA.Ethernet KRL 3.1.....	17
4.3	Installation via smarHMI.....	17
4.3.1	Installing or updating KUKA.Ethernet KRL 3.1.....	17
4.3.2	Uninstalling KUKA.Ethernet KRL 3.1.....	18
5	Configuration.....	20
5.1	Network connection via the KLI of the robot controller.....	20
6	Programming.....	21
6.1	Configuring an Ethernet connection.....	21
6.1.1	XML structure for connection properties.....	21
6.1.2	XML structure for data reception.....	24
6.1.3	XML structure for data transmission.....	26
6.1.4	Configuration according to the XPath schema.....	27
6.2	Functions for data exchange.....	28
6.2.1	Programming tips.....	29
6.2.2	Initializing and clearing a connection.....	30
6.2.3	Opening and closing a connection.....	32
6.2.4	Sending data.....	33
6.2.5	Reading out data.....	35
6.2.6	Deleting data memories.....	37

6.2.7	EKI_STATUS: Structure for function-specific return values.....	38
6.2.8	Configuration of event messages.....	39
6.2.9	Reception of complete XML data records.....	40
6.2.10	Processing incomplete data records.....	41
6.2.11	EKI_CHECK(): Checking functions for errors.....	41
7	Configuration and program examples.....	43
7.1	Integrating the server program and examples.....	43
7.1.1	Server program user interface.....	44
7.1.2	Communication parameters in the server program.....	45
7.2	“BinaryFixed” example.....	46
7.3	“BinaryStream” example.....	48
7.4	“XmlTransmit” example.....	49
7.5	“XmlServer” example.....	50
7.6	“XmlCallback” example.....	51
8	Diagnosis.....	55
8.1	Displaying diagnostic data.....	55
9	Messages.....	56
9.1	Error protocol (EKI logbook).....	56
9.2	Information about the messages.....	56
9.3	System messages from module: EthernetKRL (EKI).....	56
9.3.1	EKI00002.....	56
9.3.2	EKI00003.....	58
9.3.3	EKI00006.....	60
9.3.4	EKI00007.....	61
9.3.5	EKI00009.....	62
9.3.6	EKI00010.....	64
9.3.7	EKI00011.....	65
9.3.8	EKI00012.....	67
9.3.9	EKI00013.....	68
9.3.10	EKI00014.....	69
9.3.11	EKI00015.....	72
9.3.12	EKI00016.....	74
9.3.13	EKI00017.....	77
9.3.14	EKI00018.....	81
9.3.15	EKI00019.....	82
9.3.16	EKI00020.....	84
9.3.17	EKI00021.....	88
9.3.18	EKI00022.....	90
9.3.19	EKI00023.....	91
9.3.20	EKI00024.....	92
9.3.21	EKI00027.....	93
9.3.22	EKI00512.....	94
9.3.23	EKI00768.....	96
9.3.24	EKI01024.....	97
9.3.25	EKI01280.....	98
9.3.26	EKI01536.....	101
9.3.27	EKI01792.....	102

9.3.28	EKI02048.....	103
10	Appendix.....	104
10.1	Extended XML structure for connection properties.....	104
10.2	Increasing the memory.....	104
10.3	Deactivating message output and message logging.....	105
10.4	Command reference.....	106
10.4.1	Initializing, opening, closing and clearing a connection.....	106
10.4.2	Sending data.....	107
10.4.3	Writing data.....	108
10.4.4	Reading data.....	109
10.4.5	Checking a function for errors.....	113
10.4.6	Clearing, locking, unlocking and checking a data memory.....	113
11	KUKA Service.....	116
11.1	Requesting support.....	116
11.2	KUKA Customer Support.....	116
	Index	124

1 Introduction

1.1 Target group

This documentation is aimed at users with the following knowledge and skills:

- Advanced KRL programming skills
- Advanced knowledge of the robot controller system
- Advanced knowledge of XML
- Advanced knowledge of networks



For optimal use of our products, we recommend that our customers take part in a course of training at KUKA College. Information about the training program can be found at www.kuka.com or can be obtained directly from our subsidiaries.

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

- Documentation for the manipulator
- Documentation for the robot controller
- Operating and programming instructions for the System Software
- Instructions for options and accessories
- Parts catalog on storage medium

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

These warnings are relevant to safety and **must** be observed.



DANGER

These warnings mean that it is certain or highly probable that death or severe injuries **will** occur, if no precautions are taken.



WARNING

These warnings mean that death or severe injuries **may** occur, if no precautions are taken.



CAUTION

These warnings mean that minor injuries **may** occur, if no precautions are taken.

NOTICE

These warnings mean that damage to property **may** occur, if no precautions are taken.



These warnings contain references to safety-relevant information or general safety measures.
These warnings do not refer to individual hazards or individual precautionary measures.

This warning draws attention to procedures which serve to prevent or remedy emergencies or malfunctions:

SAFETY INSTRUCTION

The following procedure must be followed exactly!

Procedures marked with this warning **must** be followed exactly.

Notices

These notices serve to make your work easier or contain references to further information.



Tip to make your work easier or reference to further information.

1.4 Terms used

Term	Description
Data stream	Continuous sequences of data records of which the end cannot be foreseen in advance. The individual data records may be of any fixed type. The amount of data records per unit of time (data rate) may vary. Only sequential access to the data is possible.
EKI	Ethernet KRL interface
EOS	End of stream (end string) String that indicates the end of a data record
Ethernet	Ethernet is a data network technology for local area networks (LANs). It allows data to be exchanged between the connected devices in the form of data frames.
FIFO	Methods used to process a data memory
LIFO	<ul style="list-style-type: none"> • First In First Out: the elements saved first are taken first from the memory. • Last In First Out: the elements saved last are taken first from the memory.
KLI	KUKA Line Interface Line bus for the integration of the system in the customer network
KR C	KUKA Robot Controller KR C is the KUKA robot controller
KRL	KUKA Robot Language KRL is the KUKA robot programming language.
smarHMI	Smart human-machine interface KUKA smarHMI is the user interface of the KUKA system software.
Socket	Software interface that links IP addresses to port numbers.

Term	Description
TCP/IP	Transmission Control Protocol Protocol of the data exchange between devices of a network. TCP constitutes a virtual channel between two sockets in a network connection. Data can be transmitted on this channel in both directions.
UDP/IP	User Datagram Protocol Connectionless protocol for the data exchange between the devices of a network
IP	Internet Protocol The Internet Protocol is used to define subnetworks by means of physical MAC addresses.
XML	Extensible Markup Language Standard for creating machine-readable and human-readable documents in the form of a specified tree structure.
XPath	XML Path Language Language used to write and read sections of an XML document

1.5 Trademarks

.NET Framework is a trademark of Microsoft Corporation.

Windows is a trademark of Microsoft Corporation.

1.6 Licenses

The KUKA license conditions and the license conditions of the open-source software used can be found in the following folders:

- Under .\LICENSE on the data storage medium with the installation files of the KUKA software
- Under D:\KUKA_OPT*Option package name*\LICENSE after installation on the robot controller
- In the license folder under the name of the option package in the **Options** catalog after installation in WorkVisual



Further information about open-source licenses can be requested from the following address: opensource@kuka.com

2 Product description

2.1 Overview

Functions

KUKA.Ethernet KRL 3.1 is an add-on option package with the following functions:

- Data exchange via the EKI
- Receiving XML data from an external system
- Sending XML data to an external system
- Receiving binary data from an external system
- Sending binary data to an external system

Features

- Robot controller and external system as a client or server
- Configuration of connections via XML-based configuration file
- Configuration of “event messages”
- Monitoring of connections by a ping on the external system
- Reading and writing data from the Submit interpreter
- Reading and writing data from the robot interpreter

Communication

Data are transmitted via the TCP/IP protocol. It is possible to use the UDP/IP protocol, but not recommended (connectionless network protocol, e.g. no data loss detection).

2.2 Configuration of an Ethernet connection

The Ethernet connection is configured via an xml file. A configuration file must be defined for each connection in the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller. The configuration is read in when initializing a connection.

Ethernet connections can be created and operated by the robot interpreter or Submit interpreter. The channels can be used crosswise, e.g. a channel opened in the Submit interpreter can also be operated by the robot interpreter.

The deletion of a connection can be linked to robot interpreter and Submit interpreter actions or system actions.

If used crosswise, certain guidelines must be observed to avoid an unexpected program response.

(>>> [6.2.1 "Programming tips" Page 29](#))

2.2.1 Behavior in the event of a lost connection

The following properties and functions of the EKI ensure the received data can be processed reliably:

- A connection is automatically closed when reaching the limit of a data memory.
- A connection is automatically closed if a data reception error occurs.

- The data memories continue to be read out with the connection closed.
- If a connection is lost, it can be restored without any influence on the saved data.
- A lost connection can be indicated, for example, by a flag.
- The error message for the error which caused a lost connection can be displayed on the smartHMI.

2.2.2 Monitoring a connection

A connection can be monitored by a ping on the external system (<ALIVE.../> element in the connection configuration)

A flag or output can be set in the event of a successful connection, depending on the configuration. The output or flag is set as long as the ping is regularly sent and the connection to the external system is active. The output or flag is deleted if the connection to the external system is aborted.

2.3 Data exchange

Overview

The robot controller can receive data from an external system as well as send data to an external system via EKI.

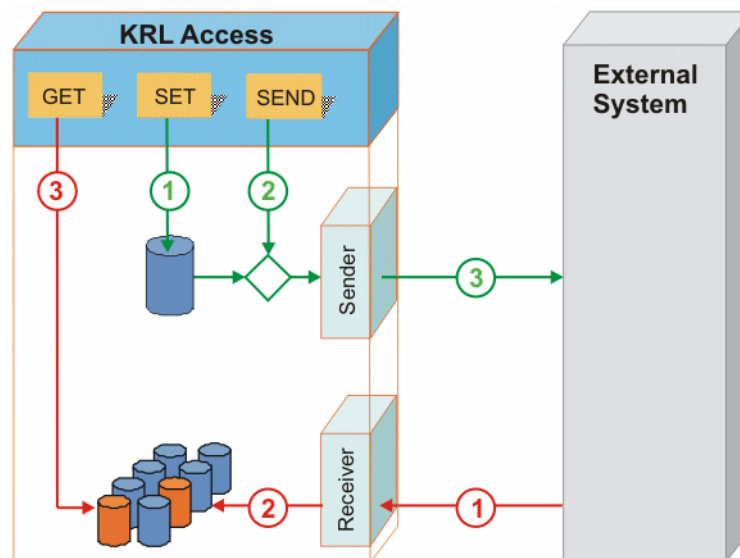


Fig. 2-1: System overview

Data reception

Basic sequence (marked in red) (>>> Fig. 2-1):

1. The external system sends data which are transmitted via a protocol and received by the EKI.
2. The data are stored in a structured manner in a data memory.
3. The data are accessed from a KRL program in a structured manner. KRL instructions are used to read the data and copy them into KRL variables.

Data transmission

Basic sequence (marked in green) (>>> *Fig. 2-1*):

1. KRL instructions are used to write the data in a data memory in a structured manner.
2. A KRL instruction is used to read the data out of the memory.
3. EKI sends the data to the external system via a protocol.



It is possible to send data directly without first storing the data in a memory.

2.4 Saving data

Description

All data received are automatically saved and, in this way, are available to KRL. XML and binary data are treated differently when saving them.

Each data memory is implemented as a memory stack. The individual memories are read out in FIFO or LIFO mode.

XML data

The received data are extracted and stored type-specifically in different memories (one memory per value).

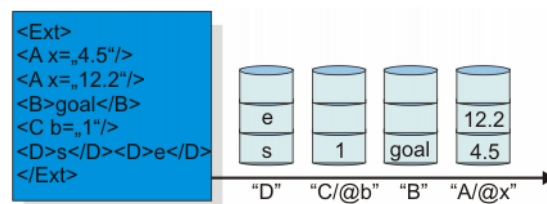


Fig. 2-2: XML data memory

Binary data

The received data are not extracted or interpreted. Only one memory exists for a connection in binary mode.

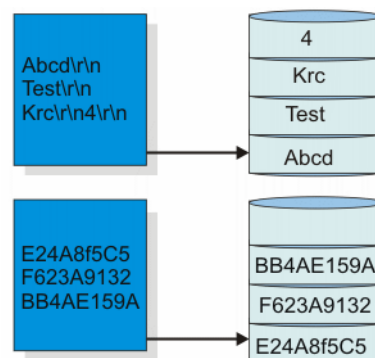


Fig. 2-3: Binary data memory

Read-out methods

Data elements are taken out of the memory in the order in which they were stored there (FIFO). The reverse method, in which the data element stored last in the memory is taken out first, can be configured (LIFO).

Each memory is assigned a common maximum limit for the data which can be saved. If the limit is exceeded, the Ethernet connection is immediately closed to prevent the reception of further data. The data currently received are still saved and the memory is increased by "1". The memories can still be further processed. The connection can be re-opened via the EKI_OPEN() function.

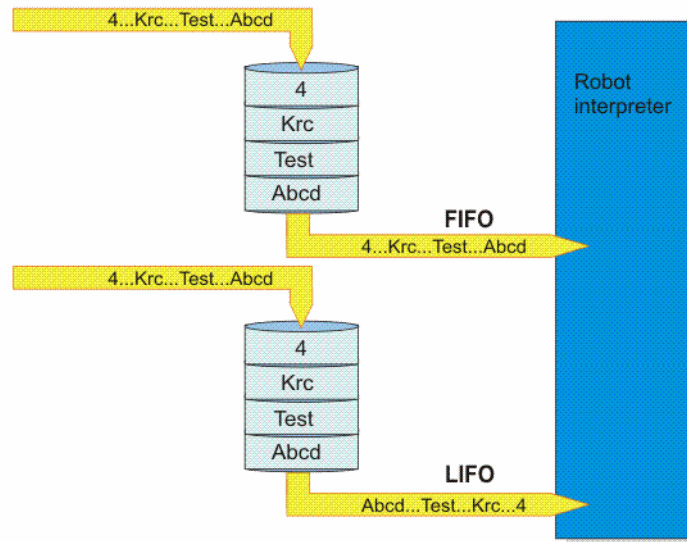


Fig. 2-4: Read-out method overview

2.5 Client-server mode

The robot controller and external system are connected as a client and server. The external system may be the client or server. The number of active connections is limited to 16.

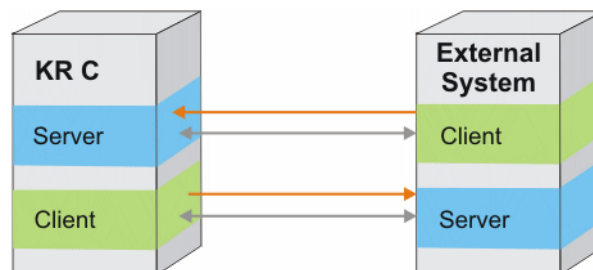


Fig. 2-5: Client-server mode

If the EKI is configured as a server, only an individual client can connect to the server. If several connections are required, several EKI servers should also be created. It is possible to operate several clients and servers simultaneously within the EKI.

2.6 Protocol types

The transmitted data can be packed in different formats.

The following formats are supported:

- Freely configurable XML structure
- Binary data set of fixed length
- Variable binary data set with end string

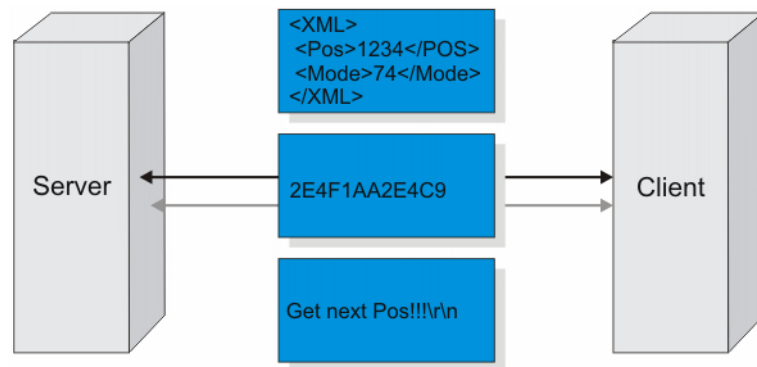


Fig. 2-6: Protocol types

2.7 Event messages

The following events can be signaled by setting an output or flag:

- Connection is active.
- An individual XML element has arrived.
- A complete XML structure or complete binary data set has arrived.

(>>> [6.2.8 "Configuration of event messages" Page 39](#))

2.8 Error treatment

KUKA.Ethernet KRL 3.1 provides functions for data exchange between the robot controller and an external system.

Each of these functions returns values. These return values can be queried and evaluated in the KRL program.

The following values are returned, depending on the function:

- Error number
- Number of elements still in the memory after the access.
- Number of elements read out of the memory
- Information on whether a connection exists
- Time stamp of the data element taken from the memory

(>>> [6.2.7 "EKI_STATUS: Structure for function-specific return values" Page 38](#))

A message is generated for each error on the smartHMI and in the EKI logbook. The automatic generation of messages can be deactivated.

2.9 Intended use

Use

The KUKA.Ethernet KRL 3.1 option package is intended for the data exchange between the robot controller and an external system via Ethernet in order to program the run time of the Submit interpreter or robot interpreter.

Misuse

Any use or application deviating from the intended use is deemed to be misuse and is not allowed. The manufacturer cannot be held liable for any resulting damage. The risk lies entirely with the user.

Examples of such misuse include:

- Communication via Ethernet is not a complete substitute for a field bus system and is not intended for operation of the Automatic External interface.
- KUKA.Ethernet KRL 3.1 does not guarantee deterministic time behavior. In other words, it cannot be assumed that called data are available at a constant time delay.
- KUKA.Ethernet KRL 3.1 is not suitable for establishing cyclic communication at the real-time cycle rate of the robot controller.

3 Safety

This documentation contains safety instructions which refer specifically to the option package described here.

The fundamental safety information for the industrial robot can be found in the “Safety” chapter of the Operating and Programming Instructions for System Integrators or the Operating and Programming Instructions for End Users.



The “Safety” chapter in the operating and programming instructions of the system software must be observed. Death to persons, severe injuries or considerable damage to property may otherwise result.

4 Installation

The option package can either be installed on the robot controller via the smartHMI or via WorkVisual.

4.1 System requirements

Robot controller

Hardware:

- KR C4

Software:

- KUKA System Software 8.6

Laptop/PC

Software:

- WorkVisual 6.0

The requirements for installation of WorkVisual are contained in the WorkVisual documentation.

4.2 Installation via WorkVisual

4.2.1 Installing or updating KUKA.Ethernet KRL 3.1

Description

The option package is installed in WorkVisual and added to the project. During project deployment, the option package is automatically installed on the robot controller.



It is advisable to archive all relevant data before updating a software package.

Precondition

- User group “Expert”
- T1 or T2 mode
- No program is selected.
- Network connection between PC and robot controller
- The option package is available as a KOP file.

Procedure

1. Install the KUKA.Ethernet KRL 3.1 option package in WorkVisual.
2. Load the active project from the robot controller.
3. Insert the KUKA.Ethernet KRL 3.1 option package into the project.
4. Configure the option package in WorkVisual as required.
5. Deploy the project from WorkVisual to the robot controller and activate it.
6. The request for confirmation *Do you want to activate the project [...]?* is displayed on the smartHMI. The active project is overwritten during

activation. If no relevant project will be overwritten: Answer the query with **Yes**.

7. An overview with the changes and a request for confirmation are displayed on the smartHMI. Answer this with **Yes**. The option package is installed and the robot controller carries out a reboot.



Information about procedures in WorkVisual is contained in the WorkVisual documentation.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.2.2 Uninstalling KUKA.Ethernet KRL 3.1

Description



The option package is uninstalled via WorkVisual.

It is advisable to archive all relevant data before uninstalling a software package.

Precondition

- User group "Expert"
- T1 or T2 mode
- No program is selected.
- Network connection between PC and robot controller

Procedure

1. Load the project from the robot controller.
2. Remove the KUKA.Ethernet KRL 3.1 option package from the project. A window with modifications is displayed.
3. Deploy the project from WorkVisual to the robot controller and activate it.
4. Answer the request for confirmation *Do you want to activate the project [...]*? on the smartHMI with **Yes**.
5. An overview with the changes and a request for confirmation are displayed on the smartHMI. Answer this with **Yes**. The option package is uninstalled and the robot controller carries out a reboot.



Information about procedures in WorkVisual is contained in the WorkVisual documentation.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.3 Installation via smartHMI

4.3.1 Installing or updating KUKA.Ethernet KRL 3.1



It is advisable to archive all relevant data before updating a software package.

Precondition

- User rights: Function group **General configuration**
But at least the user group "Expert"
- T1 or T2 mode
- No program is selected.
- USB stick with the option package (KOP file)

NOTICE

We recommend using a KUKA USB stick. Data may be lost if a stick from a different manufacturer is used.

Procedure

1. Connect the USB stick to the robot controller or smartPAD.
2. In the main menu, select **Start-up > Additional software**.
3. Press **New software**: The entry KUKA.Ethernet KRL 3.1 must be displayed in the **Name** column and drive **E:** or **K:** in the **Path** column.
If not, press **Refresh**.
4. If the specified entries are now displayed, continue with step 5.
Otherwise, the path from which the software is to be installed must be configured first:
 - a. Press the **Configure** button.
 - b. Select a line in the **Installation paths for options** area.
Note: If the line already contains a path, this path will be overwritten.
 - c. Press **Path selection**. The available drives are displayed.
 - d. If the stick is connected to the robot controller: Select **E:**.
If the stick is connected to the smartPAD: **K:** instead of **E:**
 - e. Press **Save**. The **Installation paths for options** area is displayed again. It now contains the new path.
 - f. Mark the line with the new path and press **Save** again.
5. Activate the check mark at KUKA.Ethernet KRL 3.1 and press **Install**.
Confirm the installation query with **OK**.
6. The request for confirmation *Do you want to activate the project [...]?* is displayed. The active project is overwritten during activation. If no relevant project will be overwritten: Answer the query with **Yes**.
7. An overview with the changes and a request for confirmation are displayed. Answer this with **Yes**. The option package is installed and the robot controller carries out a reboot.
8. Remove the stick.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.3.2 Uninstalling KUKA.Ethernet KRL 3.1



It is advisable to archive all relevant data before uninstalling a software package.

Precondition

- User rights: Function group **General configuration**
But at least the user group "Expert"
- T1 or T2 mode
- No program is selected.

Procedure

1. In the main menu, select **Start-up > Additional software**.
2. Activate the check mark at KUKA.Ethernet KRL 3.1 and press **Uninstall**. Answer the request for confirmation with **Yes**.
3. Answer the request for confirmation *Do you want to activate the project [...]?* with **Yes**.
4. An overview with the changes and a request for confirmation are displayed. Answer this with **Yes**. The option package is uninstalled and the robot controller carries out a reboot.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

5 Configuration

5.1 Network connection via the KLI of the robot controller

A network connection must be established via the KLI of the robot controller in order to exchange data via Ethernet.



Further information about KLI network configuration can be found in the Operating and Programming Instructions for System Integrators for the system software.

The following Ethernet interfaces are available as options at the customer interface of the robot controller, depending on the specification:

- Interface X66 (1 slot)
- Interface X67.1-3 (3 slots)



Further information on the Ethernet interfaces can be found in the operating or assembly instructions for the robot controller.

6 Programming

6.1 Configuring an Ethernet connection

An Ethernet connection is configured via an xml file. A configuration file must be defined for each connection in the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller.



xml files are case-sensitive. Upper/lower case must be taken into consideration.

The name of the xml file is also the access key in KRL.

Example: ... \EXT.xml → EKI_INIT("EXT")

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL></EXTERNAL>
    <INTERNAL></INTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <ELEMENTS></ELEMENTS>
  </RECEIVE>
  <SEND>
    <ELEMENTS></ELEMENTS>
  </SEND>
</ETHERNETKRL>
```

Section	Description
<CONFIGURATION> ... </CONFIGURATION>	Configuration of the connection parameters between the external system and the EKI (>>> 6.1.1 "XML structure for connection properties" Page 21)
<RECEIVE> ... </RECEIVE>	Configuration of the reception structure received by the robot controller (>>> 6.1.2 "XML structure for data reception" Page 24)
<SEND> ... </SEND>	Configuration of the transmission structure sent by the robot controller (>>> 6.1.3 "XML structure for data transmission" Page 26)

6.1.1 XML structure for connection properties

Description

The settings for the external system are defined in the section <EXTERNAL> ... </EXTERNAL>:

Element	Description
TYPE	Defines whether the external system is to communicate as a server or client with the robot controller (optional) <ul style="list-style-type: none"> • Server: external system is a server.

Element	Description
	<ul style="list-style-type: none"> • Client: external system is a client. <p>Default value: Server</p>
IP	<p>IP address of the external system if it is defined as a server (TYPE = server)</p> <p>The IP address is ignored if TYPE = client.</p>
PORT	<p>Port number of the external system if it is defined as a server (TYPE = server)</p> <ul style="list-style-type: none"> • 1 ... 65534 <p>Note: When selecting the port, it must be ensured that it is not being used by other services, e.g. by the operating system. Otherwise, no connection can be established via this port.</p> <p>The port number is ignored if TYPE = client.</p>

The settings for the EKI are defined in the section <INTERNAL> ... </INTERNAL>:

Element	Attribute	Description
ENVIRONMENT	—	<p>Link the deletion of the connection to actions (optional)</p> <ul style="list-style-type: none"> • Program: deletion after actions of the robot interpreter <ul style="list-style-type: none"> – Reset the program. – Deselect program. – Reconfigure I/Os. • Submit: deletion after actions of the Submit interpreter <ul style="list-style-type: none"> – Cancel Submit interpreter. – Reconfigure I/Os. • System: deletion after system actions <ul style="list-style-type: none"> – Reconfigure I/Os. <p>Default value: Program</p>
BUFFERING	Mode	<p>Method used to process all data memories (optional)</p> <ul style="list-style-type: none"> • FIFO: First In First Out • LIFO: Last In First Out <p>Default value: FIFO</p>
	Limit	<p>Maximum number of data elements which can be stored in a data memory (optional)</p> <ul style="list-style-type: none"> • 1 ... 512 <p>Default value: 16</p>
BUFSIZE	Limit	<p>Maximum number of bytes which can be received without being interpreted (optional)</p> <ul style="list-style-type: none"> • 1 ... 65,534 bytes <p>Default value: 16,384 bytes</p>

Element	Attribute	Description
TIMEOUT	Connect	Time until the attempt to establish a connection is aborted (optional) Unit: ms <ul style="list-style-type: none">• 0 ... 65,534 ms Default value: 2,000 ms
ALIVE	Set_Out	Sets an output or a flag for a successful connection (optional) Number of the output: <ul style="list-style-type: none">• 1 ... 4096 Number of the flag: <ul style="list-style-type: none">• 1 ... 1024 The output or flag is set as long as a connection to the external system is active. The output or flag is deleted if the connection to the external system is aborted.
	Set_Flag	
	Ping	Interval for sending a ping in order to monitor the connection to the external system (optional) <ul style="list-style-type: none">• 1 ... 65,534 s
IP	_____	IP address of the EKI if it is defined as a server (EXTERNAL/TYPE = client) The IP address is ignored if EXTERNAL/TYPE = server.
PORT	_____	Port number of the EKI if it is defined as a server (EXTERNAL/TYPE = client) <ul style="list-style-type: none">• 54600 ... 54615 The port number is ignored if EXTERNAL/TYPE = server.
PROTOCOL	_____	Transmission protocol (optional) <ul style="list-style-type: none">• TCP• UPD Default value: TCP It is recommended to always use the TCP/IP protocol.
MESSAGES	Logging	Deactivates the writing of messages in the EKI logbook (optional). <ul style="list-style-type: none">• warning: warning messages and error messages are logged.• error: only error messages are logged.• disabled: logging is deactivated. Default value: error
	Display	Deactivates message output on smartHMI (optional). <ul style="list-style-type: none">• error: message generation is active.

Element	Attribute	Description
		<ul style="list-style-type: none"> disabled: message output is deactivated. Default value: error
	(>>> 10.3 "Deactivating message output and message logging" Page 105)	

Example

```

<CONFIGURATION>
  <EXTERNAL>
    <IP>172.1.10.5</IP>
    <PORT>60000</PORT>
    <TYPE>Server</TYPE>
  </EXTERNAL>
  <INTERNAL>
    <ENVIRONMENT>Program</ENVIRONMENT>
    <BUFFERING Mode="FIFO" Limit="10"/>
    <BUFFSIZE Limit="16384"/>
    <TIMEOUT Connect="60000"/>
    <ALIVE Set_Out="666" Ping="200"/>
    <IP>192.1.10.20</IP>
    <PORT>54600</PORT>
    <PROTOCOL>TCP</PROTOCOL>
    <MESSAGES Logging="error" Display="disabled"/>
  </INTERNAL>
</CONFIGURATION>

```

6.1.2 XML structure for data reception

Description

The configuration depends on whether XML data or binary data are received.

- An XML structure has to be defined for the reception of XML data:
<XML> ... </XML>
- Raw data have to be defined for the reception of binary data: <RAW> ... </RAW>

Attributes in the elements of the XML structure <XML> ... </XML>:

Element	Attribute	Description
ELEMENT	Tag	Name of the element The XML structure for data reception is defined here (XPath). (>>> 6.1.4 "Configuration according to the XPath schema" Page 27)
ELEMENT	Type	Data type of the element <ul style="list-style-type: none"> STRING REAL INT BOOL FRAME

Element	Attribute	Description
		<p>Note: Optional if the tag is used only for event messages. In this case no memory capacity is reserved for the element.</p> <p>Event flag example: <ELEMENT Tag="Ext" Set_Flag="56"/></p>
ELEMENT	Set_Out	<p>Sets an output or flag after receiving the element (optional)</p> <p>Number of the output:</p> <ul style="list-style-type: none"> • 1 ... 4096 <p>Number of the flag:</p> <ul style="list-style-type: none"> • 1 ... 1024 <p>Note: If an output or flag is set by Set_Out/ Set_Flag, the associated system variable \$OUT[]/\$FLAG[] must be reset again in the program code.</p>
	Set_Flag	
ELEMENT	Mode	<p>Method used to process a data record in the data memory</p> <ul style="list-style-type: none"> • FIFO: First In First Out • LIFO: Last In First Out <p>Only relevant if individual data records are to be treated differently than configured under BUFFERING for the EKI.</p>

Attributes for the element in the raw data <RAW> ... </RAW>:

Element	Attribute	Description
ELEMENT	Tag	Name of the element
ELEMENT	Type	<p>Data type of the element</p> <ul style="list-style-type: none"> • BYTE: Binary data set of fixed length • STREAM: Binary data set with variable end strings and variable length
ELEMENT	Set_Out	<p>Sets an output or flag after receiving the element (optional)</p> <p>Number of the output:</p> <ul style="list-style-type: none"> • 1 ... 4096 <p>Number of the flag:</p> <ul style="list-style-type: none"> • 1 ... 1024 <p>Note: If an output or flag is set by Set_Out/ Set_Flag, the associated system variable \$OUT[]/\$FLAG[] must be reset again in the program code.</p>
	Set_Flag	
ELEMENT	EOS	<p>End string of an elementary piece of information (only relevant if Type = "STREAM")</p> <ul style="list-style-type: none"> • ASCII encoding: 1 ... 32 characters • Alternative end is separated by means of the "I" character.

Element	Attribute	Description
		Examples: <ul style="list-style-type: none"> • <ELEMENT ... EOS="123,134,21"/> • <ELEMENT ... EOS="123,134,21 13,10"/>
ELEMENT	Size	Fixed size of information if Type = "BYTE" <ul style="list-style-type: none"> • 1 ... 3600 bytes

Examples

```

<RECEIVE>
  <XML>
    <ELEMENT Tag="Ext/Str" Type="STRING"/>
    <ELEMENT Tag="Ext/Pos/XPos" Type="REAL" Mode="LIFO"/>
    <ELEMENT Tag="Ext/Pos/YPos" Type="REAL"/>
    <ELEMENT Tag="Ext/Pos/ZPos" Type="REAL"/>
    <ELEMENT Tag="Ext/Temp/Cpu" Type="REAL" Set_Out="1"/>
    <ELEMENT Tag="Ext/Temp/Fan" Type="REAL" Set_Flag="14"/>
    <ELEMENT Tag="Ext/Integer/AState" Type="INT"/>
    <ELEMENT Tag="Ext/Integer/BState" Type="INT"/>
    <ELEMENT Tag="Ext/Boolean/CState" Type="BOOL"/>
    <ELEMENT Tag="Ext/Frames/Frame1" Type="FRAME"/>
    <ELEMENT Tag="Ext/Attributes/@A1" Type="STRING"/>
    <ELEMENT Tag="Ext/Attributes/@A2" Type="INT"/>
    <ELEMENT Tag="Ext" Set_Flag="56"/>
  </XML>
</RECEIVE>

```

```

<RECEIVE>
  <RAW>
    <ELEMENT Tag="RawData" Type="BYTE" Size="1408"
      Set_Flag="14"/>
  </RAW>
</RECEIVE>

```

```

<RECEIVE>
  <RAW>
    <ELEMENT Tag="MyStream" Type="STREAM" EOS="123,134,21"
      Size="836" Set_Flag="14"/>
  </RAW>
</RECEIVE>

```

6.1.3 XML structure for data transmission

Description

The configuration depends on whether XML data or binary data are sent.

- An XML structure has to be defined for the transmission of XML data: <XML> ... </XML>



For transmission, the XML structure is created in the sequence in which it is configured.

- The transmission of binary data is implemented directly in the KRL programming. No configuration has to be specified.

Attribute in the elements of the XML structure <XML> ... </XML>:

Attribute	Description
Tag	Name of the element The XML structure for data transmission is defined here (XPath).

Example

```
<SEND>
<XML>
<ELEMENT Tag="Robot/Data/ActPos/@X"/>
<ELEMENT Tag="Robot/Data/ActPos/@Y"/>
<ELEMENT Tag="Robot/Data/ActPos/@Z"/>
<ELEMENT Tag="Robot/Data/ActPos/@A"/>
<ELEMENT Tag="Robot/Data/ActPos/@B"/>
<ELEMENT Tag="Robot/Data/ActPos/@C"/>
<ELEMENT Tag="Robot/Status"/>
<ELEMENT Tag="Robot/Mode"/>
<ELEMENT Tag="Robot/Complex/Tickcount"/>
<ELEMENT Tag="Robot/RobotType/Robot/Type"/>
</XML>
</SEND>
```

6.1.4 Configuration according to the XPath schema

Description

If XML is used to exchange data, it is necessary for the exchanged XML documents to be structured in the same way. KUKA.Ethernet KRL uses the XPath schema to write and read the XML documents.

The following cases are to be distinguished for XPath:

- Writing and reading elements
- Writing and reading attributes

Element notation

- Saved XML document for data transmission:

```
<Robot>
  <Mode>...</Mode>
  <RobotLamp>
    <GrenLamp>
      <LightOn>...</LightOn>
    </GrenLamp>
  </RobotLamp>
</Robot>
```

- Configured XML structure for data transmission:

```
<SEND>
<XML>
  <ELEMENT Tag="Robot/Mode" />
  <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
</XML>
<SEND />
```

Attribute notation

- Saved XML document for data transmission:

```
<Robot>
  <Data>
    <ActPos X="...">
    </ActPos>
    <LastPos A="..." B="..." C="..." X="..." Y="..."
Z="...">
    </LastPos>
  </Data>
</Robot>
```

- Configured XML structure for data transmission:

```
<SEND>
  <XML>
    <ELEMENT Tag="Robot/Data/LastPos/@X" />
    <ELEMENT Tag="Robot/Data/LastPos/@Y" />
    ...
    <ELEMENT Tag="Robot/Data/ActPos/@X" />
  </XML>
</SEND />
```

6.2 Functions for data exchange

EKI provides functions for data exchange between the robot controller and an external system.

Exact descriptions of the functions can be found in the appendix.

(>>> [10.4 "Command reference" Page 106](#))

Initializing, opening, closing and clearing a connection
EKI_STATUS = EKI_Init(CHAR[])
EKI_STATUS = EKI_Open(CHAR[])
EKI_STATUS = EKI_Close(CHAR[])
EKI_STATUS = EKI_Clear(CHAR[])
Sending data
EKI_STATUS = EKI_Send(CHAR[], CHAR[], INT)
Writing data
EKI_STATUS = EKI_SetReal(CHAR[], CHAR[], REAL)
EKI_STATUS = EKI_SetInt(CHAR[], CHAR[], INT)
EKI_STATUS = EKI_SetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_SetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_SetString(CHAR[], CHAR[], CHAR[])
Reading data
EKI_STATUS = EKI_GetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])
EKI_STATUS = EKI_GetInt(CHAR[], CHAR[], INT)
EKI_STATUS = EKI_GetIntArray(CHAR[], CHAR[], INT[])
EKI_STATUS = EKI_GetReal(CHAR[], CHAR[], REAL)

Reading data
EKI_STATUS = EKI_GetRealArray(CHAR[], CHAR[], REAL[])
EKI_STATUS = EKI_GetString(CHAR[], CHAR[], CHAR[])
EKI_STATUS = EKI_GetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])
Checking a function for errors
EKI_CHECK(EKI_STATUS, EKIMsgType, CHAR[])
Clearing, locking, unlocking and checking a data memory
EKI_STATUS = EKI_Clear(CHAR[])
EKI_STATUS = EKI_ClearBuffer(CHAR[], CHAR[])
EKI_STATUS = EKI_Lock(CHAR[])
EKI_STATUS = EKI_Unlock(CHAR[])
EKI_STATUS = EKI_CheckBuffer(CHAR[], CHAR[])

6.2.1 Programming tips

- The following points should be observed if a connection is created in the Submit interpreter:
 - The <ENVIRONMENT> element must be used in the connection configuration to specify that the channel concerned is a Submit channel.
 - An open channel in the Submit interpreter can also be addressed by the robot interpreter.
 - If the Submit interpreter is deselected, the connection is automatically deleted by means of the configuration.
- If the Submit interpreter and robot interpreter access a channel in parallel, the following points must be observed:
 - If data are read via EKI_Get...(), the contents of this memory are deleted. For this reason, it is not sensible to read data using the Submit interpreter and robot interpreter in parallel.
 - The Submit interpreter and robot interpreter can write to the same channel using EKI_Set...(). In this case, the calls can reciprocally overwrite the data if the same target element is accessed without calling EKI_Send() in between.
 - The execution times in different interpreters may vary relative to one another, resulting in unexpected behavior in the case of competing write or read access.



It is advisable to use a separate channel for each transmission and receiving task.



The description of transmission and receiving also applies if KUKA.MultiSubmitInterpreter is used.



EKI instructions are executed in the advance run!

- If an EKI instruction is to be executed in the main run, instructions must be used which trigger an advance run stop, e.g. WAIT SEC.

- Since each access to the memory consumes time, it is recommended to call up large amounts of data with the field access functions `EKI_Get...Array()`.
- A maximum of 512 array elements can be accessed via the function `EKI_Get...Array()`. In KRL, larger arrays can be created, e.g. `my-Frame[1000]`, but only a maximum of 512 elements are readable.
- There are various possibilities of waiting for data:
 - By setting a flag or an output, it can be indicated that a certain data element or a complete data record has been received (`Set_Flag` or `Set_Out` element in the XML structure for data reception).
Example: Interrupt of the KRL program by `WAIT FOR $FLAG[x]` ([>>> 7.6 "XmlCallback" example" Page 51](#))
 - The `EKI_CheckBuffer()` function can be used to check cyclically whether the memory contains new data elements.

6.2.2 Initializing and clearing a connection

Description

A connection must be created and initialized with the `EKI_Init()` function. The connection configuration specified in the function is read in. A connection can be created both in the robot interpreter and in the Submit interpreter.

A connection can be deleted again in the robot or Submit interpreter using the `EKI_Clear()` function. The deletion of a connection can additionally be linked to robot interpreter and Submit interpreter actions or system actions. (Configurable via the `<ENVIRONMENT>` element in the connection configuration)

“Program” configuration

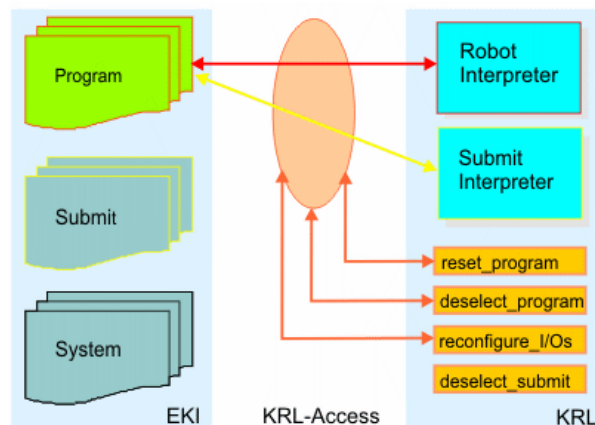


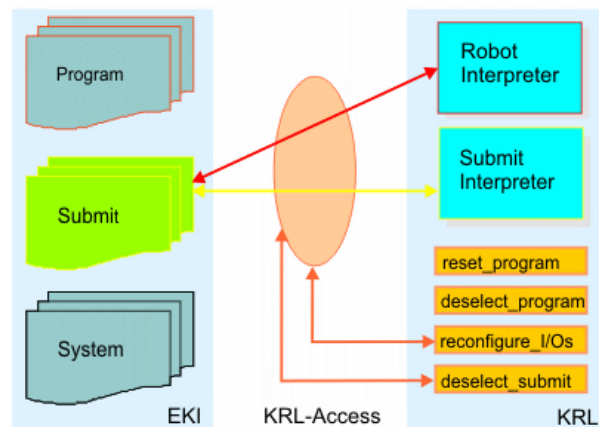
Fig. 6-1: “Program” connection configuration

With this configuration, a connection is deleted after the following actions:

- Reset program.
- Cancel program.
- Reconfigure I/Os.



When reconfiguring the I/Os, the driver is reloaded and all initializations are deleted.

“Submit” configuration**Fig. 6-2: “Submit” connection configuration**

With this configuration, a connection is deleted after the following actions:

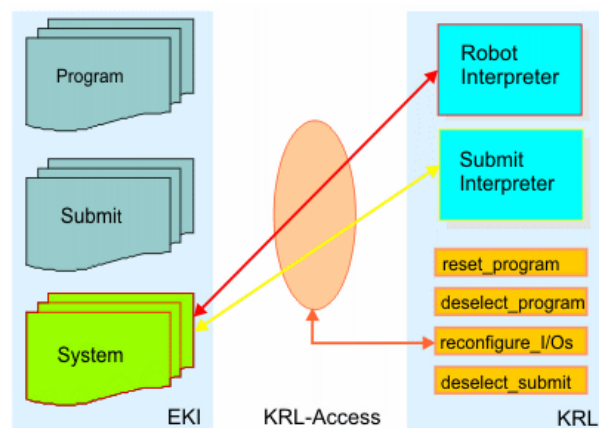
- Cancel Submit interpreter.
- Reconfigure I/Os.



If KUKA.MultiSubmitInterpreter is used, the connection can only be terminated after the system Submit interpreter has been deselected. Submit interpreters that are available in addition to the system Submit interpreter have no effect on the termination of the connection.



When reconfiguring the I/Os, the driver is reloaded and all initializations are deleted.

“System” configuration**Fig. 6-3: “System” connection configuration**

With this configuration, a connection is deleted after the following actions:

- Reconfigure I/Os.



When reconfiguring the I/Os, the driver is reloaded and all initializations are deleted.

6.2.3 Opening and closing a connection

Description

The connection to the external system is established by means of a KRL program. Most KRL programs are structured as follows:

```

1 DEF Connection()
2 ...
3 RET=EKI_Init("Connection")
4 RET=EKI_Open("Connection")
5 ...
6 Write data, send data or get received data
7 ...
8 RET=EKI_Close("Connection")
9 RET=EKI_Clear("Connection")
10 ...
11 END

```

Line	Description
3	EKI_Init() initializes the channel used by the EKI to connect to the external system.
4	EKI_Open() opens the channel.
6	KRL instructions used to write data in the memory, send data or access received data
8	EKI_Close() closes the channel.
9	EKI_Clear() clears the channel.

It should be taken into account during programming whether the EKI is configured as a server or client.

Server mode

EKI_Open() sets the EKI (= server) to a listening state if the external system is configured as a client. The server waits for the connection request of a client without interruption of the program run. If the <TIMEOUT Connect="..."/> element is not assigned data in the configuration file, the server waits until a client requests a connection.

A connection request by a client is indicated by access to the EKI or by an event message, e.g. via the <ALIVE SET_OUT="..."/> element.

An event flag or output has to be programmed, e.g. WAIT FOR \$OUT[...], if the program run is to be interrupted as long as the server waits for the connection request.



It is recommended not to use EKI_Close() in server mode. In server mode, the channel is closed from the external client.

Client mode

EKI_Open() interrupts the program run until the connection to the external system is active if the external system is configured as a server.

EKI_Close() closes the connection to the external server.

6.2.4 Sending data

Description

Depending on the configuration and programming, the following data can be sent with `EKI_Send()`:

For communication by means of XML structure:

- Complete XML structure
- Part of an XML structure:
- Random character string of variable length

For communication by means of raw data:

- For binary data sets of fixed length (attribute `Type = "BYTE"`): Random character string of fixed length
 - The size of the character string (in bytes) must correspond exactly to the configured attribute `Size`. If it is exceeded, an error message is generated. If it is not reached, a warning message is generated.
 - Binary data sets of fixed length must be read into the KRL program with `CAST_TO()`. Only data of `REAL` type (4 bytes) are legible, not `Double`.



Detailed information on the `CAST_TO()` command can be found in the CREAD/CWRITE documentation.

- For binary data sets with variable end strings (attribute `Type = "STREAM"`): Random character string of variable length
The end string is sent automatically.

Examples

Sending XML data:

Sending a complete XML structure:

- Saved XML structure for data transmission:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_Send("Channel_1", "Robot")
```

- Sent XML structure:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

Sending part of the XML structure:

- Saved XML structure for data transmission:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
```

```
<Status>12345678</Status>
</Robot>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_Send("Channel_1","Robot/ActPos")
```

- Sent XML structure:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
</Robot>
```

Directly sending a character string:

- Saved XML structure for data transmission (not used in the case of direct transmission):

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_Send("Channel_1","<POS><XPOS>1</XPOS></POS>")
```

- Sent string:

```
<POS><XPOS>1</XPOS></POS>
```

Sending binary data:

Sending a binary data record of fixed length (10 bytes):

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
</RAW>
```

- Programming:

```
DECL EKI_STATUS RET
CHAR Bytes[10]
OFFSET=0
CAST_TO(Bytes[],OFFSET,91984754,913434.2,TRUE,"X")
RET=EKI_Send("Channel_1",Bytes[])
```

- Sent data:

```
"x?{ ? _I X"
```

Sending a binary data record with variable end strings and variable length:

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="STREAM" EOS="65,66" />
</RAW>
```

- Programming:

```
DECL EKI_STATUS RET
CHAR Bytes[64]
Bytes[]="Stream ends with:"
RET=EKI_Send("Channel_1",Bytes[])
```

- Sent data:

```
"Stream ends with:AB"
```

Sending a binary data record with a maximum character limit and with end strings and variable length:

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="STREAM" EOS="65,66" />
</RAW>
```

- Programming:

```
DECL EKI_STATUS RET
CHAR Bytes[64]
Bytes[]="Stream ends with:"
RET=EKI_Send("Channel_1",Bytes[],6)
```

- Sent data:

```
"StreamAB"
```

6.2.5 Reading out data



In order to read out data, the corresponding KRL variables have to be initialized, e.g. by the assignment of values.

Description

XML and binary data are treated differently when saved and read out:

- XML data are extracted by the EKI and stored type-specifically in different memories. It is possible to access each saved value individually. All EKI_Get...() access functions can be used to read out XML data.
- Binary data records are not interpreted by the EKI and stored together in a memory.

The EKI_GetString() access function must be used to read a binary data record out of a memory. Binary data records are read out of the memory as strings.

Binary data records of fixed length must be divided into individual variables again in the KRL program with CAST_FROM().



Detailed information on the CAST_FROM() command can be found in the CREAD/CWRITE documentation.

Examples

Reading XML data:

XML structure for data reception:

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Sensor/Message" Type="STRING" />
    <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
  </XML>
</RECEIVE>
```

Programming:

```
; Declaration
INT i
DECL EKI_STATUS RET
CHAR valueChar[256]
BOOL valueBOOL
; Initialization
FOR i=(1) TO (256)
  valueChar[i]=0
ENDFOR
valueBOOL=FALSE
RET=EKI_GetString("Channel_1", "Sensor/Message", valueChar[])
RET=EKI_GetBool("Channel_1", "Sensor/Status/IsActive", value-
BOOL)
```

XML document assigned with sensor data (received data):

```
<Sensor>
  <Message>Example message</Message>
  <Status>
    <IsActive>1</IsActive>
  </Status>
</Sensor>
```

Reading a binary data record of fixed length (10 bytes):

- Configured raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
  </RAW>
</RECEIVE>
```

- Programming:

```
; Declaration
INT i
INT OFFSET
DECL EKI_STATUS RET
CHAR Bytes[10]
INT valueInt
REAL valueReal
BOOL valueBool
CHAR valueChar[1]
; Initialization
FOR i=(1) TO (10)
  Bytes[i]=0
ENDFOR
OFFSET=0
valueInt=0
valueBool=FALSE
valueReal=0
```

```
valueChar[1]=0
RET=EKI_GetString("Channel_1", "Buffer", Bytes[])
OFFSET=0
CAST_FROM(Bytes[],OFFSET,valueReal,valueInt,value-
Char[],valueBool)
```

Reading out a binary data record with end string:

- Configured raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="Buffer" Type="STREAM" EOS="13,10" />
  </RAW>
</RECEIVE>
```

- Programming:

```
; Declaration
INT i
DECL EKI_STATUS RET
CHAR Bytes[64]
; Initialization
FOR i=(1) TO (64)
  Bytes[i]=0
ENDFOR
RET=EKI_GetString("Channel_1", "Buffer", Bytes[])
```

6.2.6 Deleting data memories

Description

A distinction is to be made between the following cases when deleting data memories:

- Deletion with EKI_Clear(): Deleting all data memories and terminating the Ethernet connection
- Deletion with EKI_ClearBuffer(): Deleting specific data memories without terminating the Ethernet connection

EKI_ClearBuffer()

The data memory that can be deleted with the command EKI_ClearBuffer() depends on whether communication occurs by means of raw data or an XML structure.

For communication by means of raw data:

- The data memory for received data can be deleted.

For communication by means of XML structure:

- The data memory can be selectively deleted by entering the XPath expression of the element from the configured XML structure. This applies for both received and sent data.



XML data are extracted by the EKI and stored type-specifically in different memories. When deleting individual memories, it must be ensured that no data that belong together are lost.

Examples

For communication by means of raw data:

Deleting the data memory for received data:

- Configured raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="RawData" Type="BYTE" Size="1408"
Set_Flag="14"/>
  </RAW>
</RECEIVE>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_ClearBuffer("Channel_1","RawData")
```

For communication by means of XML structure:**Deleting the data memory for the tag <Flag>:**

- Configured XML structure for data reception:

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Ext/Activ/Value" Type="REAL"/>
    <ELEMENT Tag="Ext/Activ/Flag" Type="BOOL"/>
    <ELEMENT Tag="Ext/Activ/Flag/Message" Type="STRING"/>
  </XML>
</RECEIVE>
```

The data memory for the tag <Message> is also deleted since this tag is subordinate to the tag <Flag>.

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_ClearBuffer("Channel_1","Ext/Activ/Flag")
```

6.2.7 EKI_STATUS: Structure for function-specific return values**Description**

Each EKI function returns function-specific values. EKI_STATUS is the global structure variable to which these values are written.

Syntax

```
GLOBAL STRUC EKI_STATUS INT Buff, Read, Msg_No, BOOL
Connected, INT Counter
```

Explanation of the syntax

Element	Description
Buff	Number of elements still in the memory after access.
Read	Number of elements read out of the memory
Msg_No	Error number of the error that occurred during a function call or data reception. If automatic message output has been deactivated, EKI_CHECK() can be used to read out the error number and display the error message on the smartHMI.
Connected	Indicates whether a connection exists

Element	Description
	<ul style="list-style-type: none"> • TRUE = connection present • FALSE = connection interrupted
Counter	<p>Time stamp for received data packets</p> <p>Data packets arriving in the memory are numbered consecutively in the order in which they are stored in the memory.</p> <p>If individual data are read, the <i>Counter</i> structure element is assigned the time stamp of the data packet from which the data element originates.</p> <p>(>>> 6.2.10 "Processing incomplete data records" Page 41)</p>

Return values

The following elements of the EKI_STATUS structure are assigned data, depending on the function:

Function	Buff	Read	Msg_No	Connected	Counter
EKI_Init()	✗	✗	✓	✗	✗
EKI_Open()	✗	✗	✓	✓	✗
EKI_Close()	✗	✗	✓	✗	✗
EKI_Clear()	✗	✗	✓	✗	✗
EKI_Send()	✗	✗	✓	✓	✗
EKI_Set...()	✗	✗	✓	✓	✗
EKI_Get...()	✓	✓	✓	✓	✓
EKI_ClearBuffer()	✗	✗	✓	✓	✗
EKI_Lock()	✗	✗	✓	✓	✗
EKI_UnLock()	✗	✗	✓	✓	✗
EKI_CheckBuffer()	✓	✗	✓	✓	✓

6.2.8 Configuration of event messages

The following events can be signaled by setting an output or flag:

- Connection is active.
- An individual XML element has arrived.
- A complete XML structure or complete binary data set has arrived.

Event output

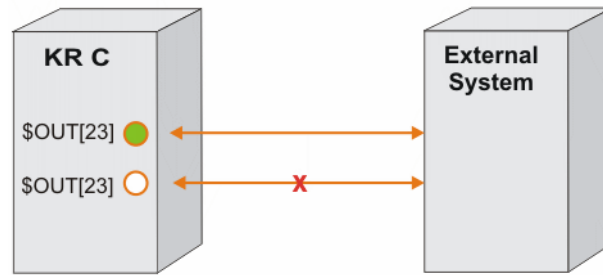


Fig. 6-4: Event output (active connection)

\$OUT[23] is set as long as the connection to the external system is active. \$OUT[23] is reset when the connection is no longer active.



The connection can be restored only with the EKI_OPEN() function.

Event flag

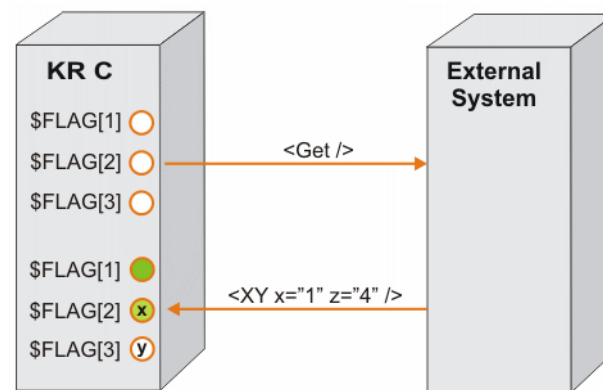


Fig. 6-5: Event flag (complete XML structure)

The XML structure `<XY />` contains the “XY/@x” and “XY/@z” data elements. \$FLAG[1] is set since the complete XML structure has arrived. \$FLAG[2] is set since the “x” element is contained in “XY”. \$FLAG[3] is not set since the “y” element has not been transferred.

Example

(>>> [7.6 “XmlCallback” example” Page 51](#))

6.2.9 Reception of complete XML data records

Description

The EKI_Get...() access functions are disabled until all data of an XML data record are in the memory.

If LIFO is configured and two or more XML data records arrive directly in succession, it is no longer ensured that a data record can be fetched in a non-fragmented condition from the memory. It may, for example, be the case that the data of the second data record are already stored in the memory although the first data record has not yet been completely processed. The data record available in the KRL is inconsistent since, in LIFO mode, the data saved last are always accessed first.

To prevent the fragmentation of data records in LIFO mode, the processing of newly received data must be disabled until all data belonging together have been fetched from the memory.

Example

```

...
RET=EKI_Lock("MyChannel")
RET=EKI_Get...()
RET=EKI_Get...()
...
RET=EKI_Get...()
RET=EKI_Unlock("MyChannel")
...

```

6.2.10 Processing incomplete data records

Under certain circumstances an external system may send incomplete data records. Individual XML elements are either empty or missing entirely, with the result that data from various data packets are present in one memory layer.

If it is necessary for the data records to be of contiguous structure in KRL, the Counter structure element of the EKI_STATUS variable can be used. When EKI_Get...Array functions are used, temporally non-contiguous data are recognized by the return of Counter = 0.

6.2.11 EKI_CHECK(): Checking functions for errors**Description**

For each error, KUKA.Ethernet KRL 3.1 displays a message on the smartHMI. The automatic generation of messages can be deactivated.

(>>> *10.3 "Deactivating message output and message logging" Page 105*)

If automatic message generation has been deactivated, it is recommended to use the EKI_CHECK() function to check whether an error has occurred during execution of a function.

- The error number is read out and the corresponding message is displayed on the smartHMI.
- If a channel name is specified in EKI_CHECK(), it is checked during data reception whether errors have occurred.

(>>> *10.4.5 "Checking a function for errors" Page 113*)

The program KRC:\R1\TP\EthernetKRL\EthernetKRL_USER.SRC is called up each time EKI_CHECK() is called up. User-specific error responses can be programmed in this program.

Example

A connection is closed whenever a reception error occurs. An interrupt can be programmed as a fault service function if the Ethernet connection is terminated.

- It is defined in the XmlTransmit.XML configuration file that FLAG[1] is set in the event of a successful connection. FLAG[1] is reset if the connection is lost.

```
<ALIVE Set_Flag="1"/>
```

- The interrupt is declared and switched on in the KRL program. The interrupt program is run if FLAG[1] is reset.

```
;FOLD Define callback
  INTERRUPT DECL 89 WHEN $FLAG[1]==FALSE DO CON_ERR()
  INTERRUPT ON 89
;ENDFOLD
```

- EKI_CHECK() is used in the interrupt program to query what sort of error occurred and then re-open the connection.

```
DEF CON_ERR()
  DECL EKI_STATUS RET
  RET={Buff 0,Read 0, Msg_no 0, Connected false}
  EKI_CHECK(RET,#Quit,"XmlTransmit")
  RET=EKI_OPEN("XmlTransmit")
END
```

7 Configuration and program examples

7.1 Integrating the server program and examples

Description

A server program as well as various example configurations and example programs are contained in the scope of supply of the software package. These example configurations and programs can be used to establish communication between the server program and the robot controller.

Components	Directory
Server program • EthernetKRL_Server.exe	DOC\Example\Application
Example program in KRL • BinaryFixed.src • BinaryStream.src • XmlCallback.src • XmlServer.src • XmlTransmit.src	DOC\Example\Program
Example configurations in XML • BinaryFixed.xml • BinaryStream.xml • XmlCallBack.xml • XmlServer.xml • XmlTransmit.xml • XmlFullConfig.xml	DOC\Example\Config

Precondition

External system:

- Windows operating system with .NET Framework 3.5 or higher installed

Robot controller:

- "Expert" user group
- Operating mode T1 or T2

Procedure

1. Copy the server program onto the external system.
2. Copy the XML example configurations into the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller.
3. Copy the KRL example programs into the directory C:\KRC\ROBOTER\KRC\R1\Program of the robot controller.
4. Start the server program on the external system.
(>>> 7.1.1 "Server program user interface" Page 44)
5. Select the menu button in the server program. The **Communication Properties** window appears.
6. Set the communication parameters as required.

(>>> 7.1.2 "Communication parameters in the server program"
Page 45)

7. Close the **Communication Properties** window and select the Start button in the server program. The port used for communication is displayed in the message window.
8. Set the IP address of the external system in the desired XML file.

7.1.1 Server program user interface

Description

The server program enables the communication between an external system and the robot controller to be tested by establishing a stable connection to the robot controller.

The server program has the following functions:

- Sending and receiving data (automatically or manually)
- Displaying the data received
- Displaying the data sent

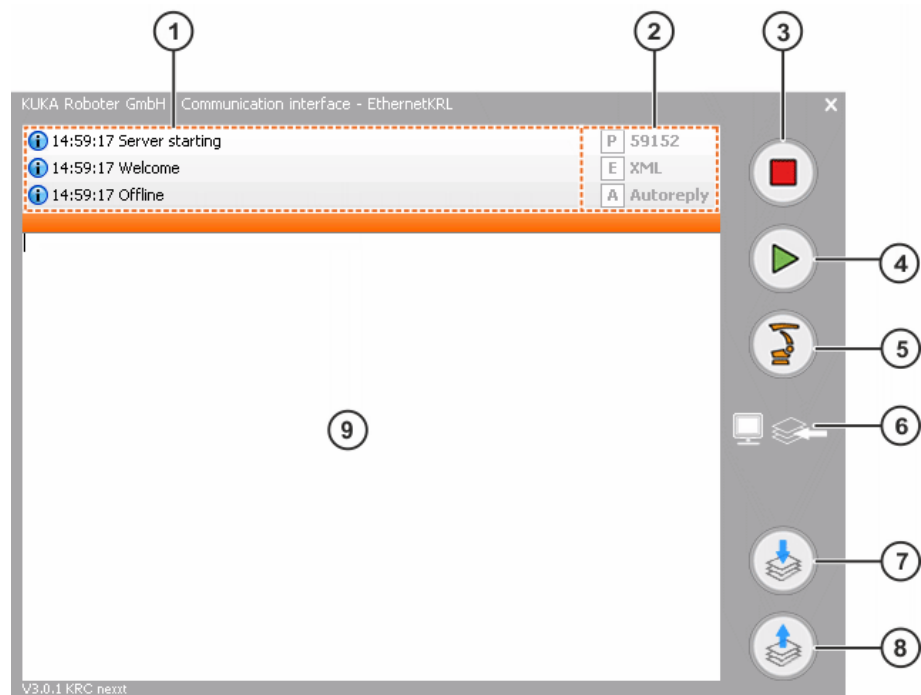


Fig. 7-1: Server program user interface

Item	Description
1	Message window
2	<p>Display of the communication parameters set</p> <p>(>>> 7.1.2 "Communication parameters in the server program" Page 45)</p> <ul style="list-style-type: none"> • P: Port number • E: Example data <ul style="list-style-type: none"> – Xml: XML data – BinaryFixed: Binary data of fixed length – BinaryStream: Variable binary data stream with end string

Item	Description
	<ul style="list-style-type: none"> • A: Communication mode <ul style="list-style-type: none"> – Autoreply: The server automatically responds to each data package received. – Manual: only manual data reception or data transmission
3	Stop button Communication with the robot controller is terminated and the server is reset.
4	Start button Data exchange between the server program and robot controller is evaluated. The first incoming connection request is linked and used as a communication adapter.
5	Menu button for setting the communication parameters (>>> 7.1.2 "Communication parameters in the server program" Page 45)
6	Display options <ul style="list-style-type: none"> • Arrow pointing to the left: the received data are displayed. (Default) • Arrow pointing to the right: the sent data are displayed.
7	Button for manual data reception
8	Button for manual data transmission
9	Display window The sent or received data are displayed, depending on the display option set.

7.1.2 Communication parameters in the server program

Description

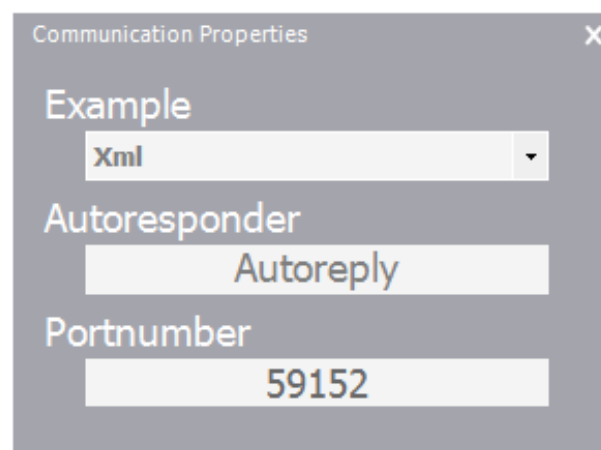


Fig. 7-2: Communication Properties window

Parameters	Description
Example	Select example data. <ul style="list-style-type: none"> • Xml: XML data • BinaryFixed: Binary data of fixed length

Parameters	Description
	<ul style="list-style-type: none"> BinaryStream: Variable binary data stream with end string <p>Default: Xml</p>
Autoresponder	<p>Select communication mode.</p> <ul style="list-style-type: none"> Autoreply: The server automatically responds to each data package received. Manual: only manual data reception or data transmission <p>Default: Autoreply</p>
Portnumber	<p>Enter the port number of the socket connection.</p> <p>The external system awaits the connection request from the robot controller at this port. A free number that is not assigned a standard service must be selected.</p> <p>Default: 59152</p> <p>Note: When selecting the port, it must be ensured that it is not being used by other services, e.g. by the operating system. Otherwise, no connection can be established via this port.</p>

7.2 “BinaryFixed” example



For communication with the robot controller, the appropriate example data must have been set in the server program; in this case “Binary-Fixed”.

The EKI is configured as a client. Only binary data records with a fixed length of 10 bytes and the element name "Buffer" can be received via this connection. The server program sends a data record. \$FLAG[1] is set when the EKI has received external data.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <RAW>
      <ELEMENT Tag="Buffer" Type="BYTE" Set_Flag="1"
Size="10" />
    </RAW>
  </RECEIVE>
  <SEND/>
</ETHERNETKRL>
```

Binary data records of fixed length must be read into and out of the KRL program with CAST_TO() and CAST_FROM(). Only data of REAL type (4 bytes) are legible, not Double.



Detailed information on the CAST_TO() and CAST_FROM() commands can be found in the CREAD/CWRITE documentation.

Program

```

1 DEF BinaryFixed( )
2 Declaration
3 INI
4 Initialize sample data
5
6 RET=EKI_Init("BinaryFixed")
7 RET=EKI_Open("BinaryFixed")
8
9 OFFSET=0
10 CAST_TO(Bytes[],OFFSET,34.425,674345,"R",TRUE)
11
12 RET=EKI_Send("BinaryFixed",Bytes[])
13
14 WAIT FOR $FLAG[1]
15 RET=EKI_GetString("BinaryFixed","Buffer",Bytes[])
16 $FLAG[1]=FALSE
17
18 OFFSET=0
19 CAST_FROM(Bytes[], OFFSET, valueReal, valueInt, value-
    Char[], valueBool)
20
21 RET=EKI_Close("BinaryFixed")
22 RET=EKI_Clear("BinaryFixed")
23 END

```

Row	Description
4	Initialization of KRL variables by the assignment of values
6	EKI_Init() initializes the channel used by the EKI to connect to the external system.
7	EKI_Open() opens the channel and connects to the server.
9, 10	CAST_TO writes the values in the Bytes[] CHAR array.
12	EKI_Send() sends the Bytes[] CHAR array to the external system.
14 ... 16	<p>\$FLAG[1] indicates the reception of the configured data element.</p> <p>EKI_GetString accesses the memory and copies the data into the Bytes[] CHAR array.</p> <p>\$FLAG[1] is reset again.</p>
18, 19	CAST_FROM reads the values out of the Bytes[] CHAR array and copies them type-specifically into the specified variables.
21	EKI_Close() closes the channel.
22	EKI_Clear() clears the channel.

7.3 “BinaryStream” example



For communication with the robot controller, the appropriate example data must have been set in the server program; in this case “BinaryStream”.

The EKI is configured as a client. Only binary data records with a maximum length of 64 bytes and the element name “Buffer” can be received via this connection. The end of the binary data record must be indicated with the end string CR, LF. FLAG[1] is set when the EKI has received this element.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <RAW>
      <ELEMENT Tag="Buffer" Type="STREAM" Set_Flag="1"
              Size="64" EOS="13,10" />
    </RAW>
  </RECEIVE>
  <SEND/>
</ETHERNETKRL>
```

Program

```
1 DEF BinaryStream( )
2 Declaration
3 INI
4 Initialize sample data
5
6 RET=EKI_Init("BinaryStream")
7 RET=EKI_Open("BinaryStream")
8
9 Bytes[]="Stream ends with CR,LF"
10
11 RET=EKI_Send("BinaryStream",Bytes[])
12
13 WAIT FOR $FLAG[1]
14 RET=EKI_GetString("BinaryStream","Buffer",Bytes[])
15 $FLAG[1]=FALSE
16
17 RET=EKI_Close("BinaryStream")
18 RET=EKI_Clear("BinaryStream")
19
20 END
```

Row	Description
4	Initialization of KRL variables by the assignment of values
6	EKI_Init() initializes the channel used by the EKI to connect to the external system.
7	EKI_Open() opens the channel and connects to the server.

Row	Description
9	The Bytes[] CHAR array is assigned data.
11	EKI_Send() sends the Bytes[] CHAR array to the external system.
13 ... 15	\$FLAG[1] indicates the reception of the configured data element. EKI_GetString reads the string in the Bytes[] CHAR array out of the memory. \$FLAG[1] is reset again.
17	EKI_Close() closes the channel.
18	EKI_Clear() clears the channel.

7.4 “XmlTransmit” example



For communication with the robot controller, the appropriate example data have to be set in the server program; in this case “Xml”.

The EKI is configured as a client. Robot data are sent and the received sensor data read out of the memory after a waiting time of 1 second.

XML file

```

<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <XML>
      <ELEMENT Tag="Sensor/Message" Type="STRING" />
      <ELEMENT Tag="Sensor/Positions/Current/@X" Type="REAL" />
      <ELEMENT Tag="Sensor/Positions/Before/X" Type="REAL" />
      <ELEMENT Tag="Sensor/Nmb" Type="INT" />
      <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
      <ELEMENT Tag="Sensor/Read/xyzabc" Type="FRAME" />
      <ELEMENT Tag="Sensor/Show/@error" Type="BOOL" />
      <ELEMENT Tag="Sensor/Show/@temp" Type="INT" />
      <ELEMENT Tag="Sensor/Show" Type="STRING" />
      <ELEMENT Tag="Sensor/Free" Type="INT" />
    </XML>
  </RECEIVE>
  <SEND>
    <XML>
      <ELEMENT Tag="Robot/Data/LastPos/@X" />
      <ELEMENT Tag="Robot/Data/LastPos/@Y" />
      <ELEMENT Tag="Robot/Data/LastPos/@Z" />
      <ELEMENT Tag="Robot/Data/LastPos/@A" />
      <ELEMENT Tag="Robot/Data/LastPos/@B" />
      <ELEMENT Tag="Robot/Data/LastPos/@C" />
      <ELEMENT Tag="Robot/Data/ActPos/@X" />
      <ELEMENT Tag="Robot/Status" />
      <ELEMENT Tag="Robot/Mode" />
    </XML>
  </SEND>
</ETHERNETKRL>

```

```

        <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
    </XML>
</SEND>
</ETHERNETKRL>

```

Program

```

1 DEF XmlTransmit( )
2 Declaration
3 Communicated data
4 INI
5 Initialize sample data
6
7 RET=EKI_Init("XmlTransmit")
8 RET=EKI_Open("XmlTransmit")
9
10 Write data to connection
11 Send data to external program
12 Get received sensor data
13
14 RET=EKI_Close("XmlTransmit")
15 RET=EKI_Clear("XmlTransmit")
16
17 END

```

Line	Description
5	Initialization of KRL variables by the assignment of values
7	EKI_Init() initializes the channel used by the EKI to connect to the external system.
8	EKI_Open() opens the channel and connects to the external system.
10	Writes data in the saved XML document for data transmission.
11	Sends the written XML document to the external system.
12	Reads the received sensor data out of the memory.
14	EKI_Close() closes the channel.
15	EKI_Clear() clears the channel.

7.5 “XmlServer” example

The EKI is configured as a server. \$FLAG[1] is set as long as a connection to the external system exists.



If the EKI has been configured as a server, the server program cannot be used on the external system. A simple client can be implemented with Windows HyperTerminal.

XML file

```

<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <TYPE>Client</TYPE>
    </EXTERNAL>
    <INTERNAL>
      <IP>x.x.x.x</IP>
    </INTERNAL>
  </CONFIGURATION>
</ETHERNETKRL>

```

```

        <PORT>54600</PORT>
        <ALIVE Set_Flag="1" />
    </INTERNAL>
</CONFIGURATION>
<RECEIVE>
    <XML>
        <ELEMENT Tag="Sensor/A" Type="BOOL" />
    </XML>
</RECEIVE>
<SEND>
    <XML>
        <ELEMENT Tag="Robot/B" />
    </XML>
</SEND>
</ETHERNETKRL>

```

Program

```

1 DEF XmlServer( )
2 Declaration
3 INI
4
5 RET=EKI_Init("XmlServer")
6 RET=EKI_Open("XmlServer")
7
8 ; wait until server is connected
9 wait for $FLAG[1]
10 ; wait until server is disconnected
11 wait for $FLAG[1]==FALSE
12
13 RET=EKI_Clear("XmlServer")
14 END

```

Line	Description
5	EKI_Init() initializes the channel used by the external system to connect to the EKI.
6	EKI_Open() opens the channel.
9	\$FLAG[1] is set when the external client has connected successfully to the server.
11	Since the EKI is configured as a server, the robot controller expects the channel to be closed by the external client. In this case, \$FLAG[1] is deleted.
13	EKI_Clear() clears the channel.

7.6 “XmlCallback” example



For communication with the robot controller, the appropriate example data have to be set in the server program; in this case “Xml”.

The EKI is configured as a client. Robot data are sent, sensor data received and then \$FLAG[1] awaited. \$FLAG[1] indicates that the sensor data have been read out.

It is configured in the XML file that \$FLAG[998] is set when the EKI has received all sensor data. This flag triggers an interrupt in the program. The configuration of the "Sensor" tag as event tag ensures that the sensor data are fetched only when all data are in the memories.

\$FLAG[998] is reset and \$FLAG[1] set when the sensor data have been read out.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <XML>
      <ELEMENT Tag="Sensor/Message" Type="STRING" />
      <ELEMENT Tag="Sensor/Positions/Current/@X" Type="REAL" />
      <ELEMENT Tag="Sensor/Positions/Before/X" Type="REAL" />
    </XML>
  </RECEIVE>
  <SEND>
    <XML>
      <ELEMENT Tag="Sensor/Nmb" Type="INT" />
      <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
      <ELEMENT Tag="Sensor/Read/xyzabc" Type="FRAME" />
      <ELEMENT Tag="Sensor/Show/@error" Type="BOOL" />
      <ELEMENT Tag="Sensor/Show/@temp" Type="INT" />
      <ELEMENT Tag="Sensor/Show" Type="STRING" />
      <ELEMENT Tag="Sensor/Free" Type="INT" Set_Out="998" />
      <ELEMENT Tag="Sensor" Set_Flag="998" />
    </XML>
  </SEND>
</ETHERNETKRL>
```

Program

```
1 DEF XmlCallBack( )
2 Declaration
3 Communicated data
4 INI
5 Define callback
6
7 RET=EKI_Init("XmlCallBack")
8 RET=EKI_Open("XmlCallBack")
9
10 Write data to connection
11 RET=EKI_Send("XmlCallBack","Robot")
12
13 ;wait until data read
```

```

14 WAIT FOR $FLAG[1]
15
16 RET=EKI_Close("XmlCallBack")
17 RET=EKI_Clear("XmlCallBack")
18 END
19
20 DEF GET_DATA()
21 Declaration
22 Initialize sample data
23 Get received sensor data
24 Signal read

```

Line	Description
5	Declaring and switching on the interrupt
7	EKI_Init() initializes the channel used by the EKI to connect to the external system.
8	EKI_Open() opens the channel.
10	Writes data in the saved XML document for data transmission.
11	Sends the data.
14	Waits for \$FLAG[1]. The event flag signals that all data have been read.
16	EKI_Close() closes the channel.
17	EKI_Clear() clears the channel.
20 ... 24	initialization of KRL variables by assigning values and reading out data \$FLAG[1] is set when all data have been read.

Data transmission

The XML document is assigned robot data by the KRL program and sent to the external system via the EKI.

```

<Robot>
  <Data>
    <LastPos X="..." Y="..." Z="..." A="..." B="..." C="...">
  </LastPos>
    <ActPos X="1000.12">
  </ActPos>
  </Data>
  <Status>12345678</Status>
  <Mode>ConnectSensor</Mode>
  <RobotLamp>
    <GrenLamp>
      <LightOn>1</LightOn>
    </GrenLamp>
  </RobotLamp>
</Robot>

```

Data reception

The XML document is assigned sensor data by the server program and received by the EKI.

```

<Sensor>
  <Message>Example message</Message>

```

```
<Positions>
<Current X="4645.2" />
  <Before>
    <X>0.9842</X>
  </Before>
</Positions>
<Nmb>8</Nmb>
<Status>
  <IsActive>1</IsActive>
</Status>
<Read>
  <xyzabc X="210.3" Y="825.3" Z="234.3" A="84.2" B="12.3"
    C="43.5" />
</Read>
<Show error="0" temp="9929">Taginfo in attributes</Show>
<Free>2912</Free>
</Sensor>
```

8 Diagnosis

8.1 Displaying diagnostic data

Precondition

- User rights: Function group **Diagnostic functions**

Procedure

1. In the main menu, select **Diagnosis > Diagnostic monitor**.
2. Select the **EKI (EthernetKRL)** module in the **Module** field.

Description

Name	Description
Total memory	Total available memory (bytes)
Allocated memory	Used memory (bytes)
Robot program connections	Number of connections initialized by the robot interpreter
Submit program connections	Number of connections initialized by the Submit interpreter
System connections	Number of connections initialized by the system
Ethernet connections	Number of open connections
Processing time	Maximum time required to process received data (refreshed every 5 seconds)
Warning messages	Number of warning messages
Error messages	Number of error messages



The warning and error messages are also counted if the automatic message output and message logging have been deactivated.

9 Messages

9.1 Error protocol (EKI logbook)

All error messages of the EKI are logged in a LOG file under C:\KRC\RO-BOTER\LOG\EthernetKRL.


9.2 Information about the messages

If an error has occurred when a function is called or data are received, EKI returns the error number. The error numbers are assigned a message text which is displayed on the smartHMI. If the automatic generation of messages is deactivated, the message can still be displayed on the smartHMI by means of EKI_CHECK().

The “Messages” chapter contains selected messages. It does not cover all the messages displayed in the message window.

9.3 System messages from module: EthernetKRL (EKI)

9.3.1 EKI00002

Message code	EKI00002
Message text	Out of system memory
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: Reserved memory insufficient (>>> Page 56) Solution: Increase the memory (>>> Page 57)</p> <p>Cause: Connection configuration uses too much memory (>>> Page 57) Solution: Modify the connection configuration (>>> Page 58)</p> <p>Cause: Several connections with high levels of data activated (>>> Page 58) Solution: Adjust the programming to use less memory (>>> Page 58)</p>

Cause: Reserved memory insufficient

Description

The memory reserved for Ethernet KRL is insufficient.

Checking instructions

- Check whether the memory requirement can be reduced by another type of configuration or programming.

If this is not possible, the memory can be increased in consultation with KUKA Deutschland GmbH.

Solution: Increase the memory



The memory may be increased only in consultation with KUKA Deutschland GmbH. (>>> *11 "KUKA Service" Page 116*)

Precondition

- User rights:
 - User group "Expert"



Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

- Open the file EthernetKRL.XML in the directory C:\KRC\ROBOTER\Config\User\Common.
- Enter the desired memory capacity in bytes in the <MemSize> element in the <EthernetKRL> section.

```
<EthernetKRL>
  <Interface>
    <MemSize>1048576</MemSize>
    ...
  </EthernetKRL>
```

- Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.
- Reboot the robot controller with the settings **Cold start** and **Reload files**.

Cause: Connection configuration uses too much memory

Description

The configured Ethernet connections use too much memory.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ... \EXT.xml → EKI_INIT("EXT")

Checking instructions

- Check the xml file(s) to see if the Ethernet connection can be configured so that less memory is used.

Solution: Modify the connection configuration**Description**

The connection configuration must be modified in such a way that less memory is used.

Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file has been modified offline, copy it into the intended directory and overwrite the old xml file.

Cause: Several connections with high levels of data activated**Description**

If several connections with high levels of data are activated simultaneously, this can lead to too much memory being used.


Checking instructions

- Check the programming to see if it can be modified so that less memory is used.

Solution: Adjust the programming to use less memory**Description**

The programming must be adjusted in such a way that less memory is used.

9.3.2 EKI00003

Message code	EKI00003
Message text	File access failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: xml file containing the connection configuration not present (>>> Page 59)</p> <p>Solution: Restore xml file containing the connection configuration (>>> Page 59)</p> <p>Cause: xml file containing the connection configuration not readable (>>> Page 60)</p>

Solution: Restore xml file containing the connection configuration
(>>> Page 60)

Cause: xml file containing the connection configuration not present

Description

The Ethernet connection was not initialized because no xml file is saved under the name specified in the function `EKI_Init()`.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...\EXT.xml → <code>EKI_INIT("EXT")</code>

The procedure for checking whether the xml file is present is as follows:

Precondition

- User group "Expert"
- Program is selected or open.

Checking instructions

1. Note the file name used in the function `EKI_Init()`.

```
RET = EKI_Init("file name")
```

 - *File name:* Name of the xml file with the connection configuration
2. Open the directory in which the xml files are stored and check whether an xml file with the name specified in the function `EKI_Init()` is present.

Solution: Restore xml file containing the connection configuration

Description

The xml file with the connection configuration must be restored and copied to the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL.

Precondition

- "Expert" user group
- T1, T2 or AUT mode

Procedure

- Copy the xml file with the connection configuration into the designated directory.

Cause: xml file containing the connection configuration not readable**Description**

The Ethernet connection was not initialized since the xml file named in the function `EKI_Init()` is not readable. The xml file is damaged and cannot be opened.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...\EXT.xml → <code>EKI_INIT("EXT")</code>

The procedure for checking whether the xml file is damaged is as follows:

Precondition

- User group "Expert"
- Program is selected or open.

Checking instructions

1. Note the file name used in the function `EKI_Init()`.

```
RET = EKI_Init("file name")
```

 - *File name:* Name of the xml file with the connection configuration
2. Open the directory in which the xml file is stored and check whether the xml file can be opened.

Solution: Restore xml file containing the connection configuration**Description**

The xml file with the connection configuration must be restored and copied to the directory `C:\KRC\ROBOTER\Config\User\Common\EthernetKRL`.


Precondition

- "Expert" user group
- T1, T2 or AUT mode

Procedure

- Copy the xml file with the connection configuration into the designated directory.

9.3.3 EK100006

Message code	EK100006
Message text	Interpretation of configuration failed
Message type	Error message
	

Effect	Path-maintaining EMERGENCY STOP
	Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Schema error in the xml file with the connection configuration (>>> Page 61) Solution: Eliminating errors in the xml file (>>> Page 61)

Cause: Schema error in the xml file with the connection configuration

Description

The xml file with the connection configuration cannot be read due to a schema error.

KUKA.Ethernet KRL uses the XPath schema. The syntax specified by the schema must be followed exactly. For example, no punctuation marks or structure elements may be missing.

The manner in which the elements and attributes in the xml file are written – including uppercase and lowercase letters – is specified and must be followed exactly.



Further information about the XPath schema can be found in the KUKA.Ethernet KRL documentation.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

Solution: Eliminating errors in the xml file

Description

The errors in the xml file must be eliminated.

Precondition


- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.


9.3.4 EKI00007

Message code	EKI00007
Message text	Writing of data to send failed
Message type	Error message

	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: XML structure for data transmission designed incorrectly (>>> Page 62) Solution: Designing the XML structure according to the XML documents to be sent (>>> Page 62)

Cause: XML structure for data transmission designed incorrectly

Description


	<p>The XML structure for data transmission cannot be described because the XML structure is designed according to a schema other than that of the XML document to be sent. KUKA.Ethernet KRL uses the XPath schema.</p> <div> <p>Further information about the XPath schema can be found in the KUKA.Ethernet KRL documentation.</p> </div>
---	---

Solution: Designing the XML structure according to the XML documents to be sent

Description

The XML structure for sending data must be designed following the XPath schema in accordance with the XML documents to be sent.

9.3.5 EKI00009

Message code	EKI00009
Message text	Connection not available
Message type	Error message 
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: EKI_Init() is programmed incorrectly or not at all (>>> Page 62) Solution: Program the function correctly (>>> Page 63)

Cause: EKI_Init() is programmed incorrectly or not at all

Description

The Ethernet connection was not initialized because the function EKI_Init() is programmed incorrectly or not at all.

A connection must always be created and initialized with the EKI_Init() function. The xml file containing the connection configuration specified in the function is loaded.

RET = EKI_Init(CHAR[])	
Function	Initializes a channel for Ethernet communication The following actions are performed: <ul style="list-style-type: none"> • The connection configuration is loaded. • The data memories are created. • The Ethernet connection is prepared.
Parameters	Type: CHAR Name of the channel (= name of the xml file with the connection configuration)
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_Init("Channel_1")

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

The procedure for checking whether the function is correctly programmed is as follows:

Precondition

- User group "Expert"
- Program is selected or open.

Checking instructions

1. Check whether the following line has been programmed:

```
RET = EKI_Init("file name")
```

 - *File name:* Name of the xml file with the connection configuration
2. Check whether the file name specified in the function EKI_Init() matches the name of the xml file with the connection configuration.

Solution: Program the function correctly

Description

The function must be correctly programmed.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.6 EKI00010

Message code	EKI00010
Message text	Ethernet is disconnected
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: EKI_Open() is programmed incorrectly or not at all (>>> Page 64) Solution: Program the function correctly (>>> Page 65)

Cause: EKI_Open() is programmed incorrectly or not at all

Description

The Ethernet connection is initialized but not open because the function EKI_Open() is programmed incorrectly or not at all.

RET = EKI_Open(CHAR[])	
Function	Opens an initialized channel If the EKI is configured as a client, the EKI connects to the external system (= Server). If the EKI has been configured as a server, the EKI waits for the connection request from the external system (= Client).
Parameters	Type: CHAR Name of the channel
RET	Type: EKI_STATUS Name of the channel (= name of the xml file with the connection configuration)
Example	RET = EKI_Open("Channel_1")

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ... \EXT.xml —> EKI_INIT("EXT")

The procedure for checking whether the function is correctly programmed is as follows:

Precondition

- User group "Expert"
- Program is selected or open.

Checking instructions

1. Check whether the following line has been programmed:

```
RET = EKI_Open("file name")
```

 - *File name*: Name of the xml file with the connection configuration
2. Check whether the file name specified in the function EKI_Open() matches the file name used in the function EKI_Init().

Solution: Program the function correctly

Description

The function must be correctly programmed.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.7 EKI00011

Message code	EKI00011
Message text	Ethernet connection to external system established
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: The Ethernet connection is already open with EKI_Open() (>>> Page 66) Solution: Delete overprogrammed function (>>> Page 66)

Cause: The Ethernet connection is already open with EKI_Open()**Description**

The Ethernet connection has already been opened with the function EKI_Open().

RET = EKI_Open(CHAR[])	
Function	<p>Opens an initialized channel</p> <p>If the EKI is configured as a client, the EKI connects to the external system (= Server).</p> <p>If the EKI has been configured as a server, the EKI waits for the connection request from the external system (= Client).</p>
Parameters	<p>Type: CHAR</p> <p>Name of the channel</p>
RET	<p>Type: EKI_STATUS</p> <p>Name of the channel (= name of the xml file with the connection configuration)</p>
Example	RET = EKI_Open("Channel_1")

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

The procedure for checking whether the connection is already open:

Precondition

- User group "Expert"
- Program is selected or open.

Checking instructions

- Check if the following line between the initialization and the closing of the connection is programmed multiple times:

```
RET = EKI_Open("file name")
```

- *File name:* Name of the xml file with the connection configuration

Solution: Delete overprogrammed function**Description**

The overprogrammed function must be deleted.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.8 EKI00012

Message code	EKI00012
Message text	Create server failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: IP address and/or port number for the EKI entered incorrectly or not at all (>>> Page 67) Solution: Entering the correct IP address and port number in the xml file (>>> Page 68)

Cause: IP address and/or port number for the EKI entered incorrectly or not at all

Description

The EKI is configured as a server and the external system as a client. In the xml file with the connection configuration, the IP address and/or port number for the EKI have not been entered or have been formally entered incorrectly. In order to ensure network security, it is possible to define the subnet from which EKI can be accessed.

Examples

Connection only possible via subnet 192.168.X.X to port 54600:

```
<INTERNAL>
<IP>192.168.23.1</IP>
<PORT>54600</PORT>
</INTERNAL>
```

Connection possible via all subnets configured in KLI to port 54600:

```
<INTERNAL>
<IP>0.0.0.0</IP>
<PORT>54600</PORT>
</INTERNAL>
```

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ... \EXT.xml → EKI_INIT("EXT")

Checking instructions

- In the xml file with the connection configuration, check if the IP address and port number have been entered correctly.

Solution: Entering the correct IP address and port number in the xml file


Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

9.3.9 EKI00013

Message code	EKI00013
Message text	Initialization of Ethernet parameters failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: IP address and/or port number for the external system not entered or entered incorrectly (>>> Page 68) Solution: Entering the correct IP address and port number in the xml file (>>> Page 69)

Cause: IP address and/or port number for the external system not entered or entered incorrectly

Description

The EKI is configured as a client and the external system as a server. In the xml file with the connection configuration, the IP address and/or port number for the external system have not been entered or have been formally entered incorrectly.

The IP address and port number for the external system must be entered in section <EXTERNAL> ... </EXTERNAL> of the xml file as follows:

- <IP>*IP address*</IP>

- <PORT>*Port number*</PORT>

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...\EXT.xml → EKI_INIT("EXT")

Checking instructions

- In the xml file with the connection configuration, check if the IP address and port number have been entered correctly.

Solution: Entering the correct IP address and port number in the xml file


Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

9.3.10 EKI00014

Message code	EKI00014
Message text	Ethernet connection to external system failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: Incorrect IP address and/or port number entered (>>> Page 70)</p> <p>Solution: Entering the correct IP address and port number in the xml file (>>> Page 70)</p> <p>Cause: Different subnetworks in the Realtime OS Network Adapter and KUKA.VisionTech (>>> Page 71)</p> <p>Solution: Adapt the IP address in VisionTechConfig.xml (>>> Page 71)</p> <p>Cause: Network cable defective or not connected correctly (>>> Page 71)</p> <p>Solution: Exchange network cable or connect it correctly (>>> Page 72)</p> <p>Cause: No Ethernet connection due to software error, external system (>>> Page 72)</p>

Solution: Eliminating errors in the software of the external system
(>>> Page 72)

Cause: No Ethernet connection due to hardware error, external system (>>> Page 72)

Solution: Eliminating the hardware error of the external system
(>>> Page 72)

Cause: Incorrect IP address and/or port number entered

Description

In the xml file with the connection configuration, the incorrect IP address and/or port number are entered. They do not match the external system or the EKI.

Depending on whether the external system is configured as a server or client, the IP address and port number must be entered as follows:

- section <EXTERNAL> ... </EXTERNAL>
 - <TYPE>Server</TYPE> or <TYPE> not specified: The external system is configured as a server.
 - <TYPE>Client</TYPE>: The external system is configured as a client.
 - If TYPE = Server, the IP address and port number of the external system must here be specified as:
 - <IP>IP address</IP>
 - <PORT>Port number</PORT>

If TYPE = Client, the specified connection parameters are ignored here.

- Section <INTERNAL> ... </INTERNAL>
 - If TYPE = Client, the IP address and port number of the EKI must here be specified as:
 - <IP>IP address</IP>
 - <PORT>Port number</PORT>

If TYPE = Server, the specified connection parameters are ignored here.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

Checking instructions

- In the xml file with the connection configuration, check if the IP address and port number have been entered correctly.

Solution: Entering the correct IP address and port number in the xml file

Precondition

- “Expert” user group

- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

Cause: Different subnetworks in the Realtime OS Network Adapter and KUKA.VisionTech

Description

The Ethernet connection allows KUKA.VisionTech to establish internal communication between VxWorks and Windows within the controller. The IP addresses in the Realtime OS Network Adapter and in **VisionTech-Config.xml** are not in the same subnetwork.

Configuration file

Directory	C:\KRC\Roboter\Config\User\Common\EthernetKRL
File	VisionTechConfig.xml

The procedure for checking whether the IP addresses are in the same subnetwork is as follows:

Checking instructions

1. In the main menu, select **Start-up > Network configuration**. The **Network configuration** window opens.
2. Click on **Advanced**. The window for advanced network configuration opens.
3. Select the **Internal subnets** tab. The number range is displayed in the **Shared memory network** box.
4. Open file.
5. Search for **<External>**.
The IP address can be found under **<IP>**.
6. Check whether the number ranges on the KUKA.HMI and in the file match.

Solution: Adapt the IP address in VisionTechConfig.xml

Description

The IP address must be adapted in VisionTechConfig.xml.

Procedure

1. Open the configuration file.
2. Search for **<External>**.
Enter the IP address of the Realtime OS Network Adapter at **<IP>**.

Cause: Network cable defective or not connected correctly

Description

The network cable is defective or the plug connection is incorrect.

The procedure for checking whether the network cable and connectors are correctly connected is as follows:

Checking instructions

1. Check that the network cables are correctly connected and fitted securely.
2. Interchange the network cables.

Solution: Exchange network cable or connect it correctly

Procedure

- Exchange network cable or connect it correctly.

Cause: No Ethernet connection due to software error, external system

Description

Due to an error in the software of the external system, there is not Ethernet connection.

Checking instructions

- Check external system for software errors.

Solution: Eliminating errors in the software of the external system

Description

The errors in the software of the external system must be eliminated.

Cause: No Ethernet connection due to hardware error, external system

Description

Due to an error in the hardware of the external system (i.e. loose socket connection, defective network card, etc.) there is not Ethernet connection.

Checking instructions

- Check external system for hardware errors.


Solution: Eliminating the hardware error of the external system

Description

The hardware error of the external system must be eliminated.

9.3.11 EKI00015

Message code	EKI00015
Message text	Access to empty element memory
Message type	Error message

	
Effect	<p>Path-maintaining EMERGENCY STOP</p> <p>Input of active commands (robot motions, program start) is blocked.</p>
Possible cause(s)	<p>Cause: Empty data memory when accessed using EKI_Get...() (>>> Page 73)</p> <p>Solution: Alter the program. (>>> Page 73)</p>

Cause: Empty data memory when accessed using EKI_Get...()

Description

Access to an empty memory was achieved using an EKI_Get...() function. Access to an empty memory can be prevented by polling and evaluating the corresponding return value of the access function.

EKI_STATUS is the global structure variable into which the return values of the function are written. The element `Buff` from the EKI_STATUS is relevant for the evaluation.

`Buff` contains the following value:

- Number of elements still in the memory after access.

Syntax

```
GLOBAL STRUC EKI_STATUS INT Buff, Read, Msg_No, BOOL
Connected, INT Counter
```

Example

```
...
REPEAT
    ret = EKI_GetInt("MyChannel", "Root/Number", value)
    DoSomething(value)
UNTIL (ret.Buff < 1)
...
```

Solution: Alter the program.


Precondition

- "Expert" user group
- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.12 EKI00016

Message code	EKI00016
Message text	Element not found
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: Element in XML structure for data reception configured incorrectly or not at all (>>> Page 74)</p> <p>Solution: Eliminating errors in the xml file (>>> Page 75)</p> <p>Cause: Element name in access function configured incorrectly (>>> Page 76)</p> <p>Solution: Program the function correctly (>>> Page 77)</p>

Cause: Element in XML structure for data reception configured incorrectly or not at all

Description

The element specified in an access function is not configured in the XML structure for data reception or does not match the configured element.

Access functions

Parameter 2 of an access function always specifies the element that is to be accessed.

Access functions	
EKI_STATUS =	EKI_GetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS =	EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])
EKI_STATUS =	EKI_GetInt(CHAR[], CHAR[], INT)
EKI_STATUS =	EKI_GetIntArray(CHAR[], CHAR[], INT[])
EKI_STATUS =	EKI_GetReal(CHAR[], CHAR[], REAL)
EKI_STATUS =	EKI_GetRealArray(CHAR[], CHAR[], REAL[])
EKI_STATUS =	EKI_GetString(CHAR[], CHAR[], CHAR[])
EKI_STATUS =	EKI_GetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS =	EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ... \EXT.xml → EKI_INIT("EXT")

The reception structure is configured in the <RECEIVE> ... </RECEIVE> section of the xml file. The attribute `Tag` defines the elements that can be accessed.

Checking instructions

1. Check whether the element you were attempting to access is configured in the reception structure.
2. Check whether the programmed element name matches the configured element name.

Examples

Reading XML data:

XML structure:

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Sensor/Message" Type="STRING" />
    <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
  </XML>
</RECEIVE>
```

Programming:

```
RET=EKI_GetString("Channel_1", "Sensor/Message", valueChar[])
RET=EKI_GetBool("Channel_1", "Sensor/Status/IsActive", value-
BOOL)
```

Reading a binary data record of fixed length (10 bytes):

- Raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
  </RAW>
</RECEIVE>
```

- Programming:

```
RET=EKI_GetString("Channel_1", "Buffer", Bytes[])
```

Solution: Eliminating errors in the xml file

Description

The errors in the xml file must be eliminated.

Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

Cause: Element name in access function configured incorrectly**Description**

The element name specified in an access function does not match the element name configured in the XML structure for data reception.

Access functions

Parameter 2 of an access function always specifies the element that is to be accessed.

Access functions
EKI_STATUS = EKI_GetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])
EKI_STATUS = EKI_GetInt(CHAR[], CHAR[], INT)
EKI_STATUS = EKI_GetIntArray(CHAR[], CHAR[], INT[])
EKI_STATUS = EKI_GetReal(CHAR[], CHAR[], REAL)
EKI_STATUS = EKI_GetRealArray(CHAR[], CHAR[], REAL[])
EKI_STATUS = EKI_GetString(CHAR[], CHAR[], CHAR[])
EKI_STATUS = EKI_GetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

The reception structure is configured in the <RECEIVE> ... </RECEIVE> section of the xml file. The attribute `Tag` defines the elements that can be accessed.

Checking instructions

- Check whether the programmed element name matches the configured element name.

Examples**Reading XML data:**

XML structure:

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Sensor/Message" Type="STRING" />
    <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
  </XML>
</RECEIVE>
```

Programming:

```
RET=EKI_GetString("Channel_1", "Sensor/Message", valueChar[])
RET=EKI_GetBool("Channel_1", "Sensor/Status/IsActive", value-
BOOL)
```

Reading a binary data record of fixed length (10 bytes):

- Raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
  </RAW>
</RECEIVE>
```

- Programming:

```
RET=EKI_GetString("Channel_1", "Buffer", Bytes[])
```

Solution: Program the function correctly

Description

The function must be correctly programmed.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.13 EKI00017

Message code	EKI00017
Message text	Assembly of data to send failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: XML structure for data transmission configured incorrectly (>>> Page 78) Solution: Eliminating errors in the xml file (>>> Page 79) Cause: EKI_Send() programmed incorrectly (>>> Page 79) Solution: Program the function correctly (>>> Page 81)

Cause: XML structure for data transmission configured incorrectly**Description**

The XML structure for data transmission does not match the XML document to which the XML data are to be written. It also potentially does not match the programming of the EKI_Send() function.

RET = EKI_Send(CHAR[], CHAR[], INT)	
Function	Sends data via a channel. (>>> 6.2.4 "Sending data" Page 33)
Parameter 1	Type: CHAR Name of the channel to be used for sending
Parameter 2	Type: CHAR Defines the quantity of data to be sent. For communication by means of XML structure: <ul style="list-style-type: none"> • XPath expression of the element to be transmitted from the configured XML structure for data transmission. <ul style="list-style-type: none"> – If only the root element is specified, the entire XML structure is transferred (see example 1). – If only part of the XML structure is to be transferred (i.e. the element of a subordinate level), the path to this element from the root element must be specified (see example 2). • Alternatively: Random character string of variable length which is to be transferred For communication by means of raw data: <ul style="list-style-type: none"> • Random character string which is to be transferred: <ul style="list-style-type: none"> – For binary data sets of fixed length (attribute <code>Type = "BYTE"</code>): Random character string of fixed length The size of the character string (in bytes) must correspond exactly to the configured attribute <code>Size</code>. If it is exceeded, an error message is generated. If it is not reached, a warning message is generated. – For binary data sets with variable end strings (attribute <code>Type = "STREAM"</code>): Random character string of variable length The end string is sent automatically. <p>Note: If a random character string of variable length contains an ASCII NULL character, only the portion up to this character is transferred. If this is not desired, parameter 3 must be defined accordingly.</p>
Parameter 3 (optional)	Type: INT Only relevant for sending random character strings of variable length (see parameter 2): Maximum number of characters to be sent. ASCII NULL characters are ignored. If the character string is too long, it is cut.
RET	Type: EKI_STATUS Return values of the function
Example 1	RET = EKI_Send("Channel_1", "Root")
Example 2	RET = EKI_Send("Channel_1", "Root/Test")
Example 3	RET = EKI_Send("Channel_1", MyBytes[])

RET = EKI_Send(CHAR[], CHAR[], INT)	
Example 4	RET = EKI_Send("Channel_1", MyBytes[], 6)

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...\EXT.xml → EKI_INIT("EXT")

The transmission structure is configured in the <SEND> ... </SEND> section of the xml file.

Checking instructions

1. Check the configuration of the transmission structure in the xml file.
2. Check whether the programming of the data sent matches the configuration.

Solution: Eliminating errors in the xml file

Description

The errors in the xml file must be eliminated.

Precondition

- "Expert" user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

Cause: EKI_Send() programmed incorrectly

Description

The data to be transmitted are entered incorrectly in the EKI_Send() function. They potentially do not match the configured XML structure for data transmission.

RET = EKI_Send(CHAR[], CHAR[], INT)	
Function	Sends data via a channel. (>>> 6.2.4 "Sending data" Page 33)
Parameter 1	Type: CHAR Name of the channel to be used for sending
Parameter 2	Type: CHAR Defines the quantity of data to be sent. For communication by means of XML structure: <ul style="list-style-type: none"> • XPath expression of the element to be transmitted from the configured XML structure for data transmission.

RET = EKI_Send(CHAR[], CHAR[], INT)	
	<ul style="list-style-type: none"> – If only the root element is specified, the entire XML structure is transferred (see example 1). – If only part of the XML structure is to be transferred (i.e. the element of a subordinate level), the path to this element from the root element must be specified (see example 2). • Alternatively: Random character string of variable length which is to be transferred <p>For communication by means of raw data:</p> <ul style="list-style-type: none"> • Random character string which is to be transferred: <ul style="list-style-type: none"> – For binary data sets of fixed length (attribute <code>Type = "BYTE"</code>): Random character string of fixed length The size of the character string (in bytes) must correspond exactly to the configured attribute <code>Size</code>. If it is exceeded, an error message is generated. If it is not reached, a warning message is generated. – For binary data sets with variable end strings (attribute <code>Type = "STREAM"</code>): Random character string of variable length The end string is sent automatically. <p>Note: If a random character string of variable length contains an ASCII NULL character, only the portion up to this character is transferred. If this is not desired, parameter 3 must be defined accordingly.</p>
Parameter 3 (optional)	<p>Type: INT</p> <p>Only relevant for sending random character strings of variable length (see parameter 2): Maximum number of characters to be sent.</p> <p>ASCII NULL characters are ignored. If the character string is too long, it is cut.</p>
RET	<p>Type: EKI_STATUS</p> <p>Return values of the function</p>
Example 1	RET = EKI_Send("Channel_1", "Root")
Example 2	RET = EKI_Send("Channel_1", "Root/Test")
Example 3	RET = EKI_Send("Channel_1", MyBytes[])
Example 4	RET = EKI_Send("Channel_1", MyBytes[], 6)

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

The transmission structure is configured in the `<SEND> ... </SEND>` section of the xml file.

Checking instructions

1. Check the configuration of the transmission structure in the xml file.
2. Check whether the programming of the data sent matches the configuration.

Solution: Program the function correctly

Description

The function must be correctly programmed.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.14 EKI00018

Message code	EKI00018
Message text	Send data failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: Network cable defective or not connected correctly (>>> Page 81) Solution: Exchange network cable or connect it correctly (>>> Page 82)</p> <p>Cause: No Ethernet connection due to software error, external system (>>> Page 82) Solution: Eliminating errors in the software of the external system (>>> Page 82)</p> <p>Cause: No Ethernet connection due to hardware error, external system (>>> Page 82) Solution: Eliminating the hardware error of the external system (>>> Page 82)</p>

Cause: Network cable defective or not connected correctly

Description

The network cable is defective or the plug connection is incorrect.

The procedure for checking whether the network cable and connectors are correctly connected is as follows:

Checking instructions

1. Check that the network cables are correctly connected and fitted securely.
2. Interchange the network cables.

Solution: Exchange network cable or connect it correctly

Procedure

- Exchange network cable or connect it correctly.

Cause: No Ethernet connection due to software error, external system

Description

Due to an error in the software of the external system, there is not Ethernet connection.

Checking instructions

- Check external system for software errors.

Solution: Eliminating errors in the software of the external system

Description

The errors in the software of the external system must be eliminated.

Cause: No Ethernet connection due to hardware error, external system

Description

Due to an error in the hardware of the external system (i.e. loose socket connection, defective network card, etc.) there is not Ethernet connection.

Checking instructions

- Check external system for hardware errors.


Solution: Eliminating the hardware error of the external system

Description

The hardware error of the external system must be eliminated.

9.3.15 EKI00019

Message code	EKI00019
Message text	No data to send
Message type	Error message

Effect	
	Path-maintaining EMERGENCY STOP
Possible cause(s)	Input of active commands (robot motions, program start) is blocked.
	Cause: EKI_Send() contains no data to be sent (>>> Page 83) Solution: Program the function correctly (>>> Page 84)

Cause: EKI_Send() contains no data to be sent

Description

An 'EKI_Send()' send function contains no data to be sent (parameter 2).

RET = EKI_Send(CHAR[], CHAR[], INT)	
Function	Sends data via a channel. (>>> 6.2.4 "Sending data" Page 33)
Parameter 1	Type: CHAR Name of the channel to be used for sending
Parameter 2	Type: CHAR Defines the quantity of data to be sent. For communication by means of XML structure: <ul style="list-style-type: none"> XPath expression of the element to be transmitted from the configured XML structure for data transmission. <ul style="list-style-type: none"> If only the root element is specified, the entire XML structure is transferred (see example 1). If only part of the XML structure is to be transferred (i.e. the element of a subordinate level), the path to this element from the root element must be specified (see example 2). Alternatively: Random character string of variable length which is to be transferred For communication by means of raw data: <ul style="list-style-type: none"> Random character string which is to be transferred: <ul style="list-style-type: none"> For binary data sets of fixed length (attribute <code>Type = "BYTE"</code>): Random character string of fixed length The size of the character string (in bytes) must correspond exactly to the configured attribute <code>Size</code>. If it is exceeded, an error message is generated. If it is not reached, a warning message is generated. For binary data sets with variable end strings (attribute <code>Type = "STREAM"</code>): Random character string of variable length The end string is sent automatically. Note: If a random character string of variable length contains an ASCII NULL character, only the portion up to this character is transferred. If this is not desired, parameter 3 must be defined accordingly.
Parameter 3 (optional)	Type: INT Only relevant for sending random character strings of variable length (see parameter 2): Maximum number of characters to be sent.

RET = EKI_Send(CHAR[], CHAR[], INT)	
	ASCII NULL characters are ignored. If the character string is too long, it is cut.
RET	Type: EKI_STATUS Return values of the function
Example 1	RET = EKI_Send("Channel_1", "Root")
Example 2	RET = EKI_Send("Channel_1", "Root/Test")
Example 3	RET = EKI_Send("Channel_1", MyBytes[])
Example 4	RET = EKI_Send("Channel_1", MyBytes[], 6)

Checking instructions

- In the KRL program, check whether the data to be sent are specified.

Solution: Program the function correctly

Description

The function must be correctly programmed.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.16 EKI00020

Message code	EKI00020
Message text	Mismatch in type of data
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Data type in XML structure for data reception configured incorrectly (>>> Page 85) Solution: Eliminating errors in the xml file (>>> Page 86)

Cause: Data type in access function programmed incorrectly
(>>> Page 86)

Solution: Program the function correctly (>>> Page 88)

Cause: Data type in XML structure for data reception configured incorrectly

Description

The data type specified for an element in an access function does not match the data type configured in the XML structure for data reception.

Access functions

Parameter 3 of an access function always specifies the data type of the element that is to be accessed.

Access functions
EKI_STATUS = EKI_GetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])
EKI_STATUS = EKI_GetInt(CHAR[], CHAR[], INT)
EKI_STATUS = EKI_GetIntArray(CHAR[], CHAR[], INT[])
EKI_STATUS = EKI_GetReal(CHAR[], CHAR[], REAL)
EKI_STATUS = EKI_GetRealArray(CHAR[], CHAR[], REAL[])
EKI_STATUS = EKI_GetString(CHAR[], CHAR[], CHAR[])
EKI_STATUS = EKI_GetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

The reception structure is configured in the <RECEIVE> ... </RECEIVE> section of the xml file. The attribute `Type` defines the data type of an element.

Checking instructions

- Check whether the programmed data type matches the data type of the element configured in the reception structure.

Examples

Reading XML data:

XML structure:

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Sensor/Message" Type="STRING" />
    <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
  </XML>
</RECEIVE>
```

Programming:

```
...
CHAR valueChar[256]
BOOL valueBOOL
...
RET=EKI_GetString("Channel_1", "Sensor/Message", valueChar[])
RET=EKI_GetBool("Channel_1", "Sensor/Status/IsActive", value-
BOOL)
```

Reading a binary data record of fixed length (10 bytes):

- Raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
  </RAW>
</RECEIVE>
```

- Programming:

```
...
CHAR Bytes[10]
...
RET=EKI_GetString("Channel_1", "Buffer", Bytes[])
```

Solution: Eliminating errors in the xml file

Description

The errors in the xml file must be eliminated.

Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

Cause: Data type in access function programmed incorrectly

Description

The data type specified for an element in an access function does not match the data type configured in the XML structure for data reception.

Access functions

Parameter 3 of an access function always specifies the data type of the element that is to be accessed.

Access functions
EKI_STATUS = EKI_GetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])
EKI_STATUS = EKI_GetInt(CHAR[], CHAR[], INT)

Access functions
EKI_STATUS = EKI_GetIntArray(CHAR[], CHAR[], INT[])
EKI_STATUS = EKI_GetReal(CHAR[], CHAR[], REAL)
EKI_STATUS = EKI_GetRealArray(CHAR[], CHAR[], REAL[])
EKI_STATUS = EKI_GetString(CHAR[], CHAR[], CHAR[])
EKI_STATUS = EKI_GetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

The reception structure is configured in the <RECEIVE> ... </RECEIVE> section of the xml file. The attribute `Type` defines the data type of an element.

Checking instructions

- Check whether the programmed data type matches the data type of the element configured in the reception structure.

Examples

Reading XML data:

XML structure:

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Sensor/Message" Type="STRING" />
    <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
  </XML>
</RECEIVE>
```

Programming:

```
...
CHAR valueChar[256]
BOOL valueBOOL
...
RET=EKI_GetString("Channel_1", "Sensor/Message", valueChar[])
RET=EKI_GetBool("Channel_1", "Sensor/Status/IsActive", value-
BOOL)
```

Reading a binary data record of fixed length (10 bytes):

- Raw data:

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
  </RAW>
</RECEIVE>
```

- Programming:

```
...
CHAR Bytes[10]
...
RET=EKI_GetString("Channel_1", "Buffer", Bytes[])
```

Solution: Program the function correctly

Description

The function must be correctly programmed.

Precondition

- User rights:
 - User group “Expert”




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.17 EKI00021

Message code	EKI00021
Message text	System memory insufficient with maximum data storage
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Reserved memory insufficient (>>> Page 88) Solution: Increase the memory (>>> Page 89) Cause: Connection configuration uses too much memory (>>> Page 89) Solution: Modify the connection configuration (>>> Page 90)

Cause: Reserved memory insufficient

Description

The memory reserved for Ethernet KRL is insufficient.

Checking instructions

- Check whether the memory requirement can be reduced by another type of configuration or programming.

If this is not possible, the memory can be increased in consultation with KUKA Deutschland GmbH.

Solution: Increase the memory



The memory may be increased only in consultation with KUKA Deutschland GmbH. (>>> [11 "KUKA Service" Page 116](#))

Precondition

- User rights:
 - User group "Expert"



Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Open the file EthernetKRL.XML in the directory C:\KRC\ROBOTER\Config\User\Common.
2. Enter the desired memory capacity in bytes in the <MemSize> element in the <EthernetKRL> section.

```
<EthernetKRL>
  <Interface>
    <MemSize>1048576</MemSize>
    ...
  </EthernetKRL>
```

3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.
4. Reboot the robot controller with the settings **Cold start** and **Reload files**.

Cause: Connection configuration uses too much memory

Description

The configured Ethernet connections use too much memory.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml —> EKI_INIT("EXT")

Checking instructions

- Check the xml file(s) to see if the Ethernet connection can be configured so that less memory is used.

Solution: Modify the connection configuration

Description

The connection configuration must be modified in such a way that less memory is used.


Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file has been modified offline, copy it into the intended directory and overwrite the old xml file.

9.3.18 EKI00022

Message code	EKI00022
Message text	Error while reading the configuration. XML not valid.
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Schema error in the xml file with the connection configuration (>>> Page 90) Solution: Eliminating errors in the xml file (>>> Page 91)

Cause: Schema error in the xml file with the connection configuration

Description

The xml file with the connection configuration cannot be read due to a schema error.

KUKA.Ethernet KRL uses the XPath schema. The syntax specified by the schema must be followed exactly. For example, no punctuation marks or structure elements may be missing.

The manner in which the elements and attributes in the xml file are written – including uppercase and lowercase letters – is specified and must be followed exactly.



Further information about the XPath schema can be found in the KUKA.Ethernet KRL documentation.

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

Solution: Eliminating errors in the xml file

Description

The errors in the xml file must be eliminated.


Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

9.3.19 EKI00023

Message code	EKI00023
Message text	Initialization already performed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Ethernet connection already initialized with EKI_Init() (>>> Page 91) Solution: Delete overprogrammed function (>>> Page 92)

Cause: Ethernet connection already initialized with EKI_Init()

Description

The Ethernet connection has already been initialized with the function EKI_Init().

RET = EKI_Init(CHAR[])	
Function	Initializes a channel for Ethernet communication The following actions are performed: <ul style="list-style-type: none"> • The connection configuration is loaded. • The data memories are created. • The Ethernet connection is prepared.
Parameters	Type: CHAR Name of the channel (= name of the xml file with the connection configuration)
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_Init("Channel_1")

The procedure for checking whether the connection is already initialized:

Precondition

- User group "Expert"
- Program is selected or open.

Checking instructions

- Check if the following line between the first initialization and the closing of the connection is programmed multiple times:

```
RET = EKI_Init("file name")
```

- *File name*: Name of the xml file with the connection configuration

Solution: Delete overprogrammed function

Description

The overprogrammed function must be deleted.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.20 EKI00024

Message code	EKI00024
Message text	Link to internal parameters (Port, IP) failed
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: IP address and/or port number for the EKI entered incorrectly or not at all (>>> Page 93) Solution: Entering the correct IP address and port number in the xml file (>>> Page 93)

Cause: IP address and/or port number for the EKI entered incorrectly or not at all**Description**

The EKI is configured as a server and the external system as a client. In the xml file with the connection configuration, the IP address and/or port number for the EKI have not been entered or have been formally entered incorrectly. In order to ensure network security, it is possible to define the subnet from which EKI can be accessed.

Examples

Connection only possible via subnet 192.168.X.X to port 54600:

```
<INTERNAL>
<IP>192.168.23.1</IP>
<PORT>54600</PORT>
</INTERNAL>
```

Connection possible via all subnets configured in KLI to port 54600:

```
<INTERNAL>
<IP>0.0.0.0</IP>
<PORT>54600</PORT>
</INTERNAL>
```

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

Checking instructions

- In the xml file with the connection configuration, check if the IP address and port number have been entered correctly.

Solution: Entering the correct IP address and port number in the xml file**Precondition**


- "Expert" user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

9.3.21 EKI00027

Message code	EKI00027
Message text	CHAR[] Array too small.
Message type	Error message

	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: CHAR array too small for received data (>>> Page 94) Solution: Increasing the memory size for the CHAR array (>>> Page 94)

Cause: CHAR array too small for received data

Description

The CHAR array defined in the KRL program is too small for the received data.

Checking instructions

- Check the maximum acceptable lengths for the data sets sent by the external system.

Solution: Increasing the memory size for the CHAR array

Description

The memory size for the CHAR array must be increased.

Precondition

- User rights:
 - User group "Expert"




Depending on the configuration and the system software used, higher user rights may be required.

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.22 EKI00512

Message code	EKI00512
Message text	Ethernet connection disrupted
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP

Possible cause(s)	<p>Input of active commands (robot motions, program start) is blocked.</p> <p>Cause: Network cable defective or not connected correctly (>>> Page 95)</p> <p>Solution: Exchange network cable or connect it correctly (>>> Page 95)</p> <p>Cause: No Ethernet connection due to software error, external system (>>> Page 95)</p> <p>Solution: Eliminating errors in the software of the external system (>>> Page 95)</p> <p>Cause: No Ethernet connection due to hardware error, external system (>>> Page 96)</p> <p>Solution: Eliminating the hardware error of the external system (>>> Page 96)</p>
-------------------	---

Cause: Network cable defective or not connected correctly

Description

The network cable is defective or the plug connection is incorrect.

The procedure for checking whether the network cable and connectors are correctly connected is as follows:

Checking instructions

1. Check that the network cables are correctly connected and fitted securely.
2. Interchange the network cables.

Solution: Exchange network cable or connect it correctly

Procedure

- Exchange network cable or connect it correctly.

Cause: No Ethernet connection due to software error, external system

Description

Due to an error in the software of the external system, there is not Ethernet connection.

Checking instructions

- Check external system for software errors.

Solution: Eliminating errors in the software of the external system

Description

The errors in the software of the external system must be eliminated.

Cause: No Ethernet connection due to hardware error, external system**Description**

Due to an error in the hardware of the external system (i.e. loose socket connection, defective network card, etc.) there is not Ethernet connection.


Checking instructions

- Check external system for hardware errors.

Solution: Eliminating the hardware error of the external system**Description**

The hardware error of the external system must be eliminated.

9.3.23 EKI00768

Message code	EKI00768
Message text	Ping reports no contact
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: Network cable defective or not connected correctly (>>> Page 96)</p> <p>Solution: Exchange network cable or connect it correctly (>>> Page 97)</p> <p>Cause: No Ethernet connection due to hardware error, external system (>>> Page 97)</p> <p>Solution: Eliminating the hardware error of the external system (>>> Page 97)</p>

Cause: Network cable defective or not connected correctly**Description**

The network cable is defective or the plug connection is incorrect.

The procedure for checking whether the network cable and connectors are correctly connected is as follows:

Checking instructions

1. Check that the network cables are correctly connected and fitted securely.
2. Interchange the network cables.

Solution: Exchange network cable or connect it correctly**Procedure**

- Exchange network cable or connect it correctly.

Cause: No Ethernet connection due to hardware error, external system**Description**

Due to an error in the hardware of the external system (i.e. loose socket connection, defective network card, etc.) there is not Ethernet connection.


Checking instructions

- Check external system for hardware errors.

Solution: Eliminating the hardware error of the external system**Description**

The hardware error of the external system must be eliminated.

9.3.24 EKI01024

Message code	EKI01024
Message text	Error while reading received XML data
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Sent XML document does not correspond to the XPath schema (>>> Page 97) Solution: Designing the sent XML document according to the XPath schema (>>> Page 98)

Cause: Sent XML document does not correspond to the XPath schema**Description**

The XML document sent by the external system does not correspond to the XPath schema.




Further information about the XPath schema can be found in the KUKA.Ethernet KRL documentation.

Solution: Designing the sent XML document according to the XPath schema

Description

The XML document sent by the external system must be designed according to the XPath schema.

9.3.25 EKI01280

Message code	EKI01280
Message text	Limit of element storage reached
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	<p>Cause: Memory limit reached (number of data elements per memory) (>>> Page 98) Solution: Increase the memory limit in the xml file (>>> Page 99)</p> <p>Cause: Memory limit reached (number of data elements per memory) (>>> Page 99) Solution: Cyclically checking the data memory and disabling before reaching the limit (>>> Page 100)</p> <p>Cause: EKI memory is not read (>>> Page 100) Solution: Alter the program. (>>> Page 101)</p>

Cause: Memory limit reached (number of data elements per memory)

Description

The number of data elements that can be saved in the memory is limited. This memory limit has been reached. The Ethernet connection has been closed to prevent the reception of further data.

The maximum number of elements that can be saved in the memory is defined in the xml file with the connection configuration:

- Section <INTERNAL> ... </INTERNAL>
- Element <BUFFERING ... Limit="..."/>
 - Maximum possible number: 512
- If the BUFFERING element is not configured, the default value applies:
 - <BUFFERING Mode="FIFO" Limit="16"/>

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

Checking instructions

- In the xml file with the connection configuration, check which value is set as the limit number of elements that can be saved.

Solution: Increase the memory limit in the xml file**Description**

The memory limit in the xml file can be increased.

Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

Cause: Memory limit reached (number of data elements per memory)**Description**

The number of data elements that can be saved in the memory is limited. This memory limit has been reached. The Ethernet connection has been closed to prevent the reception of further data.

The maximum number of elements that can be saved in the memory is defined in the xml file with the connection configuration:

- Section <INTERNAL> ... </INTERNAL>
- Element <BUFFERING ... Limit="..."/>
 - Maximum possible number: 512
- If the BUFFERING element is not configured, the default value applies:
 - <BUFFERING Mode="FIFO" Limit="16"/>

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ... \EXT.xml → EKI_INIT("EXT")

Checking instructions

- In the xml file with the connection configuration, check which value is set as the limit number of elements that can be saved.

Solution: Cyclically checking the data memory and disabling before reaching the limit

Description

Programming can be used to prevent the memory limit being reached. For this, cyclically check how many data elements are currently in the data memory and temporarily disable the memory before reaching the limit.

The following functions are required:

- `EKI_STATUS = EKI_CheckBuffer(CHAR[], CHAR[])`
Checks how much data is still in the memory. The memory is not changed.
- `EKI_STATUS = EKI_Lock(CHAR[])`
Disables the processing of received data. The data can no longer be stored in the memory.
- `EKI_STATUS = EKI_Unlock(CHAR[])`
Enables the processing of received data. The data are stored in the memory again.

`EKI_STATUS` is the global structure variable into which the return values of the function are written. The element `Buff` from the `EKI_STATUS` is relevant for the evaluation.

`Buff` contains the following value:

- Number of elements still in the memory after access.

Syntax

```
GLOBAL STRUC EKI_STATUS INT Buff, Read, Msg_No, BOOL
Connected, INT Counter
```

Example

```
...
DECL INT limit
DECL EKI_STATUS ret
...
limit = 16
ret = EKI_CheckBuffer("MyChannel", "MyData")
IF(ret.Buff >= limit - 1) THEN
    ret = EKI_Lock("MyChannel")
ELSE
    ret = EKI_Unlock("MyChannel")
ENDIF
```

Cause: EKI memory is not read

Description

The external system writes data to the EKI memory. If these data are not called, the EKI memory fills up. To prevent loss of data, the connection is closed when the EKI memory is full. The TCP protocol detects the closed connection and interrupts data exchange. The maximum number of data elements has been written to the EKI memory without them being read.

The procedure for checking whether the EKI memory in the KRL program is read is as follows:

Precondition

- Program is selected or open.

Checking instructions

- Check whether the data is read via EKI_Get...().


Solution: Alter the program.**Precondition**

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

9.3.26 EKI01536

Message code	EKI01536
Message text	Received string too long
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Character string sent by the external system too long (>>> Page 101) Solution: Adjusting the programming of the external system (>>> Page 102)

Cause: Character string sent by the external system too long**Description**

A character string sent by the external system exceeds the maximum permissible length. Up to 3600 characters are permitted.

Checking instructions


- Check the length of the data sent by the external system.

Solution: Adjusting the programming of the external system

Description

The programming of the external system must be adjusted in such a way that character strings of excessive length can no longer be sent.

9.3.27 EKI01792

Message code	EKI01792
Message text	Limit of element memory reached
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Memory limit reached (number of bytes for total memory) (>>> Page 102) Solution: Increase the memory limit in the xml file (>>> Page 103)

Cause: Memory limit reached (number of bytes for total memory)

Description

The number of bytes that can be saved in the memory is limited. This memory limit has been reached. The Ethernet connection has been closed to prevent the reception of further data.

The maximum number of bytes that can be saved in the memory is defined in the xml file with the connection configuration:

- Section <INTERNAL> ... </INTERNAL>
- Element <BUFSIZE Limit="..."/>
 - Maximum possible number: 65,534
- If the BUFSIZE element is not configured, the default value applies:
 - <BUFSIZE Limit="16384"/>

Configuration file

An xml file must be configured for every Ethernet connection. The name of the xml file is also the access key in KRL.

Directory	C:\KRC\ROBOTER\Config\User\Common\EthernetKRL
File	Example: ...EXT.xml → EKI_INIT("EXT")

Checking instructions

- Check in the xml file with the connection configuration to see which byte number is the limit.

Solution: Increase the memory limit in the xml file

Description

The memory limit in the xml file can be increased.


Precondition

- “Expert” user group
- T1, T2 or AUT mode

Procedure

1. Modify the xml file as required and save the changes.
2. If the xml file was modified offline, copy it into the intended directory and overwrite the old xml file.

9.3.28 EKI02048

Message code	EKI02048
Message text	Server time limit reached
Message type	Error message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	Cause: Software error on the external system leads to timeout (>>> Page 103) Solution: Eliminating errors in the software of the external system (>>> Page 103)

Cause: Software error on the external system leads to timeout

Description

The EKI is configured as a server and is waiting for the connection request from the external system.

A value for <TIMEOUT Connect="..."/> is entered in the xml file with the connection configuration. This time has elapsed without the external system establishing a connection to the server. The cause is a fault in the software of the external system.

Checking instructions

- Check external system for software errors.

Solution: Eliminating errors in the software of the external system

Description

The errors in the software of the external system must be eliminated.

10 Appendix

10.1 Extended XML structure for connection properties



The extended xml structure may only be used in consultation with KUKA Deutschland GmbH. (>>> [11 "KUKA Service" Page 116](#))

Description

Further properties for the EKI can be configured in the section <INTERNAL> ... </INTERNAL> of the configuration file:

Element	Attribute	Description
TIMEOUT	Receive	Time until the attempt to receive data is aborted (optional) • 0 ... 65,534 ms Default value: 0 ms
	Send	Time until the attempt to send data is aborted (optional) • 0 ... 65,534 ms Default value: 0 ms
BUFFSIZE	Receive	Size of the socket used to receive data (optional) • 1 ... 65,534 bytes Default value: Predefined by the system
	Send	Size of the socket used to send data (optional) • 1 ... 65,534 bytes Default value: Predefined by the system

10.2 Increasing the memory



The memory may be increased only in consultation with KUKA Deutschland GmbH. (>>> [11 "KUKA Service" Page 116](#))

Description

If the available memory is insufficient, it is recommended to check the programming method in KRL as well as the configuration.

- Check whether a connection is configured in such a manner that the memory is completely occupied with received data.
- Check whether several connections with high levels of data have been defined and activated.

Precondition

- User rights for editing the file:
 - Function group **File operations**
- User rights for shutdown with **Reload files**:
 - Function group **Critical configurations**

- T1, T2 or AUT mode

Procedure

1. Open the file EthernetKRL.XML in the directory C:\KRC\ROBOTER\Config\User\Common.
2. Enter the desired memory capacity in bytes in the <MemSize> element in the <EthernetKRL> section.

```
<EthernetKRL>
  <Interface>
    <MemSize>1048576</MemSize>
    ...
  </EthernetKRL>
```

3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.
4. Reboot the robot controller with the settings **Cold start** and **Reload files**.

10.3 Deactivating message output and message logging

Description

It is recommended for automatic message output on the smartHMI to be deactivated in the following cases:

- Runtime errors occur.
- The EKI is used in the Submit interpreter.

If automatic message output is deactivated, all error messages are still logged by default. If these errors or warnings are to be deliberately ignored, e.g. because logging the messages causes a high system load and slows down performance, this mechanism can likewise be deactivated.

Precondition

- User rights: Function group **File operations**
But at least the user group **"Expert"**

Procedure

1. Open the configuration file of the Ethernet connection in the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller.
2. Enter the following line in the section <INTERNAL> ... </INTERNAL> of the XML file:

```
<MESSAGES Logging="disabled" Display="disabled"/>
```

3. Save the changes and close the file.



If automatic message output is deactivated, the EKI_CHECK() function can be used to check individual EKI instructions for errors.

10.4 Command reference

10.4.1 Initializing, opening, closing and clearing a connection

RET = EKI_Init(CHAR[])	
Function	<p>Initializes a channel for Ethernet communication</p> <p>The following actions are performed:</p> <ul style="list-style-type: none"> • The connection configuration is loaded. • The data memories are created. • The Ethernet connection is prepared.
Parameters	<p>Type: CHAR</p> <p>Name of the channel (= name of the xml file with the connection configuration)</p>
RET	<p>Type: EKI_STATUS</p> <p>Return values of the function</p>
Example	RET = EKI_Init("Channel_1")

RET = EKI_Open(CHAR[])	
Function	<p>Opens an initialized channel</p> <p>If the EKI is configured as a client, the EKI connects to the external system (= Server).</p> <p>If the EKI has been configured as a server, the EKI waits for the connection request from the external system (= Client).</p>
Parameters	<p>Type: CHAR</p> <p>Name of the channel</p>
RET	<p>Type: EKI_STATUS</p> <p>Name of the channel (= name of the xml file with the connection configuration)</p>
Example	RET = EKI_Open("Channel_1")

RET = EKI_Close(CHAR[])	
Function	Closes an open channel
Parameters	<p>Type: CHAR</p> <p>Name of the channel (= name of the xml file with the connection configuration)</p>
RET	<p>Type: EKI_STATUS</p> <p>Return values of the function</p>
Example	RET = EKI_Close("Channel_1")

RET = EKI_Clear(CHAR[])	
Function	Deletes a channel as well as all associated data memory and terminates the Ethernet connection.
Parameters	<p>Type: CHAR</p> <p>Name of the channel (= name of the xml file with the connection configuration)</p>

RET = EKI_Clear(CHAR[])	
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_Clear("Channel_1")

Information about the return values of the relevant function can be found here: (>>> ["Return values" Page 39](#))

10.4.2 Sending data

RET = EKI_Send(CHAR[], CHAR[], INT)	
Function	Sends data via a channel. (>>> 6.2.4 "Sending data" Page 33)
Parameter 1	Type: CHAR Name of the channel to be used for sending
Parameter 2	Type: CHAR Defines the quantity of data to be sent. For communication by means of XML structure: <ul style="list-style-type: none"> • XPath expression of the element to be transmitted from the configured XML structure for data transmission. <ul style="list-style-type: none"> – If only the root element is specified, the entire XML structure is transferred (see example 1). – If only part of the XML structure is to be transferred (i.e. the element of a subordinate level), the path to this element from the root element must be specified (see example 2). • Alternatively: Random character string of variable length which is to be transferred For communication by means of raw data: <ul style="list-style-type: none"> • Random character string which is to be transferred: <ul style="list-style-type: none"> – For binary data sets of fixed length (attribute <code>Type = "BYTE"</code>): Random character string of fixed length The size of the character string (in bytes) must correspond exactly to the configured attribute <code>Size</code>. If it is exceeded, an error message is generated. If it is not reached, a warning message is generated. – For binary data sets with variable end strings (attribute <code>Type = "STREAM"</code>): Random character string of variable length The end string is sent automatically. <p>Note: If a random character string of variable length contains an ASCII NULL character, only the portion up to this character is transferred. If this is not desired, parameter 3 must be defined accordingly.</p>
Parameter 3 (optional)	Type: INT Only relevant for sending random character strings of variable length (see parameter 2): Maximum number of characters to be sent. ASCII NULL characters are ignored. If the character string is too long, it is cut.
RET	Type: EKI_STATUS

RET = EKI_Send(CHAR[], CHAR[], INT)	
	Return values of the function
Example 1	RET = EKI_Send("Channel_1", "Root")
Example 2	RET = EKI_Send("Channel_1", "Root/Test")
Example 3	RET = EKI_Send("Channel_1", MyBytes[])
Example 4	RET = EKI_Send("Channel_1", MyBytes[], 6)

Information about the return values of the relevant function can be found here: (>>> ["Return values" Page 39](#))

10.4.3 Writing data

RET = EKI_SetBool(CHAR[], CHAR[], BOOL)	
Function	Writes a Boolean value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: BOOL Value written in the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_SetBool("Channel_1", "Root/Activ", true)

RET = EKI_SetFrame(CHAR[], CHAR[], FRAME)	
Function	Writes a FRAME type value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: FRAME Value written in the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET= EKI_SetFrame("Channel_1", "Root/BASE", {X 0.0, Y 0.0, Z 0.0, A 0.0, B 0.0, C 0.0})

RET = EKI_SetInt(CHAR[], CHAR[], INT)	
Function	Writes an integer value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: INT Value written in the memory

RET = EKI_SetInt(CHAR[], CHAR[], INT)	
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_SetInt("Channel_1", "Root/List", 67234)

RET = EKI_SetReal(CHAR[], CHAR[], REAL)	
Function	Writes a floating point value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: REAL Value written in the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_SetReal("Channel_1", "Root/Number", 1.234)

RET = EKI_SetString(CHAR[], CHAR[], CHAR[])	
Function	Writes a string in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: CHAR String written in the memory Maximum number of characters: <ul style="list-style-type: none"> • 3,600
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_SetString("Channel_1", "Root/Message", "Hello")

Information about the return values of the relevant function can be found here: (>>> ["Return values" Page 39](#))

10.4.4 Reading data

RET = EKI_GetBool(CHAR[], CHAR[], BOOL)	
Function	Reads a Boolean value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: BOOL

RET = EKI_GetBool(CHAR[], CHAR[], BOOL)	
	Value read out of the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetBool("Channel_1", "Root/Activ", My-Bool)

RET = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])	
Function	Reads a Boolean value out of the memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: BOOL Array read out of the memory Maximum number of readable array elements: <ul style="list-style-type: none">• 512
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetBoolArray("Channel_1", "Root/Activ", MyBool[])

RET = EKI_GetInt(CHAR[], CHAR[], INT)	
Function	Reads an integer value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: INT Value read out of the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetInt("Channel_1", "Root/Numbers/One", MyInteger)

RET = EKI_GetIntArray(CHAR[], CHAR[], INT[])	
Function	Reads an integer value out of a memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR

RET = EKI_GetIntArray(CHAR[], CHAR[], INT[])	
	Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: INT Array read out of the memory Maximum number of readable array elements: • 512
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetIntArray("Channel_1", "Root/Numbers/One", MyInteger[])

RET = EKI_GetReal(CHAR[], CHAR[], REAL)	
Function	Reads a floating point value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: REAL Value read out of the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetReal("Channel_1", "Root/Position", MyReal)

RET = EKI_GetRealArray(CHAR[], CHAR[], REAL[])	
Function	Reads a floating point value out of a memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: REAL Array read out of the memory Maximum number of readable array elements: • 512
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetRealArray("Channel_1", "Root/Position", MyReal[])

RET = EKI_GetString(CHAR[], CHAR[], CHAR[])	
Function	Reads a string out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure or name of the element in the raw data
Parameter 3	Type: CHAR String read out of the memory Maximum number of characters: • 3,600
RET	Type: EKI_STATUS Return values of the function
XML example	RET = EKI_GetString("Channel_1", "Root/Message", MyChars[])
Binary example	RET = EKI_GetString("Channel_1", "Streams", MyStream[])

RET = EKI_GetFrame(CHAR[], CHAR[], FRAME)	
Function	Reads a FRAME type value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: FRAME Value read out of the memory
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetFrame("Channel_1", "Root/TCP", MyFrame)

RET = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])	
Function	Reads a FRAME type value out of a memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: FRAME Array read out of the memory

RET = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])	
	Maximum number of readable array elements: • 512
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_GetFrameArray("Channel_1", "Root/TCP", MyFrame[])

Information about the return values of the relevant function can be found here: (>>> ["Return values" Page 39](#))

10.4.5 Checking a function for errors

EKI_CHECK(EKI_STATUS, EKriMsgType, CHAR[])	
Function	Checks whether an error occurred during execution of a function: • The error number is read out and the corresponding message is displayed on the smartHMI. (see parameter 1) • Optional: If the channel name is specified, it is checked during data reception whether errors have occurred (see parameter 3)
Parameter 1	EKI_STATUS Return values of the checked function
Parameter 2	Type: ENUM Type of message displayed on the smartHMI: • #NOTIFY : Notification message • #STATE : Status message • #QUIT : Acknowledgement message • #WAITING : Wait message
Parameter 3 (optional)	Type: CHAR Name of the open channel to be checked
Example 1	EKI_CHECK(RET,#QUIT)
Example 2	EKI_CHECK(RET,#NOTIFY,"MyChannelName")

Information about the return values of the relevant function can be found here: (>>> ["Return values" Page 39](#))

10.4.6 Clearing, locking, unlocking and checking a data memory

RET = EKI_Clear(CHAR[])	
Function	Deletes a channel as well as all associated data memory and terminates the Ethernet connection.
Parameters	Type: CHAR Name of the channel (= name of the xml file with the connection configuration)
RET	Type: EKI_STATUS Return values of the function
Example	RET = EKI_Clear("Channel_1")

RET = EKI_ClearBuffer(CHAR[], CHAR[])	
Function	Clears the defined data memory without terminating the Ethernet connection (>>> 6.2.6 "Deleting data memories" Page 37)
Parameter 1	Type: CHAR Name of the channel
Parameter 2	Type: CHAR Defines the data memory to be deleted For communication by means of raw data: <ul style="list-style-type: none"> Name of the raw data to be deleted from the configured XML structure for data reception For communication by means of XML structure: <ul style="list-style-type: none"> XPATH expression of the element for which the data memory is to be deleted – from the configured XML structure for data reception or from the XML structure for data transmission
RET	Type: EKI_STATUS Return values of the function
Example 1	RET = EKI_ClearBuffer("Channel_1", "RawData")
Example 2	RET = EKI_ClearBuffer("Channel_1", "Ext/Activ/Flag")

RET = EKI_Lock(CHAR[])	
Function	Disables the processing of received data. The data can no longer be stored in the memory.
Parameter	Type: CHAR Name of the channel
RET	Type: EKI_STATUS Return values of the function

RET = EKI_Unlock(CHAR[])	
Function	Enables the processing of received data. The data are stored in the memory again.
Parameter	Type: CHAR Name of the channel
RET	Type: EKI_STATUS Return values of the function

RET = EKI_CheckBuffer(CHAR[], CHAR[])	
Function	Checks how much data is still in the memory. The memory is not changed. Additionally, the time stamp of the next data element ready to be taken from the memory is returned.
Parameter 1	Type: CHAR Name of the channel

RET = EKI_CheckBuffer(CHAR[], CHAR[])	
Parameter 2	Type: CHAR Defines the data memory to be checked For communication by means of raw data: <ul style="list-style-type: none"> Name of the raw data to be checked from the configured XML structure for data transmission For communication by means of XML structure: <ul style="list-style-type: none"> XPATH expression of the element for which the data memory is to be checked – from the configured XML structure for data reception or from the XML structure for data transmission
RET	Type: EKI_STATUS Return values of the function
Example 1	RET = EKI_CheckBuffer("Channel_1", "RawData")
Example 2	RET = EKI_CheckBuffer("Channel_1", "Ext/Activ/Flag")

Information about the return values of the relevant function can be found here: (>>> ["Return values" Page 39](#))

11 KUKA Service

11.1 Requesting support

Introduction

This documentation provides information on operation and operator control, and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Information

The following information is required for processing a support request:

- Description of the problem, including information about the duration and frequency of the fault
- As comprehensive information as possible about the hardware and software components of the overall system

The following list gives an indication of the information which is relevant in many cases:

- Model and serial number of the kinematic system, e.g. the manipulator
- Model and serial number of the controller
- Model and serial number of the energy supply system
- Designation and version of the system software
- Designations and versions of other software components or modifications
- Diagnostic package KRCDiag

Additionally for KUKA Sunrise: Existing projects including applications

For versions of KUKA System Software older than V8: Archive of the software (KRCDiag is not yet available here.)
- Application used
- External axes used

11.2 KUKA Customer Support

Availability

KUKA Customer Support is available in many countries. Please do not hesitate to contact us if you have any questions.

Argentina

Ruben Costantini S.A. (Agency)
 Luis Angel Huergo 13 20
 Parque Industrial
 2400 San Francisco (CBA)
 Argentina
 Tel. +54 3564 421033
 Fax +54 3564 428877
 ventas@costantini-sa.com

Australia

KUKA Robotics Australia Pty Ltd
45 Fennell Street
Port Melbourne VIC 3207
Australia
Tel. +61 3 9939 9656
info@kuka-robotics.com.au
www.kuka-robotics.com.au

Belgium

KUKA Automatisering + Robots N.V.
Centrum Zuid 1031
3530 Houthalen
Belgium
Tel. +32 11 516160
Fax +32 11 526794
info@kuka.be
www.kuka.be

Brazil

KUKA Roboter do Brasil Ltda.
Travessa Claudio Armando, nº 171
Bloco 5 - Galpões 51/52
Bairro Assunção
CEP 09861-7630 São Bernardo do Campo - SP
Brazil
Tel. +55 11 4942-8299
Fax +55 11 2201-7883
info@kuka-roboter.com.br
www.kuka-roboter.com.br

Chile

Robotec S.A. (Agency)
Santiago de Chile
Chile
Tel. +56 2 331-5951
Fax +56 2 331-5952
robotec@robotec.cl
www.robotec.cl

China

KUKA Robotics China Co., Ltd.
No. 889 Kungang Road
Xiaokunshan Town
Songjiang District
201614 Shanghai
P. R. China
Tel. +86 21 5707 2688
Fax +86 21 5707 2603
info@kuka-robotics.cn
www.kuka-robotics.com

Germany

KUKA Deutschland GmbH
Zugspitzstr. 140
86165 Augsburg
Germany
Tel. +49 821 797-1926
Fax +49 821 797-41 1926
Hotline.robotics.de@kuka.com
www.kuka.com

France

KUKA Automatisme + Robotique SAS
Techvallée
6, Avenue du Parc
91140 Villebon S/Yvette
France
Tel. +33 1 6931660-0
Fax +33 1 6931660-1
commercial@kuka.fr
www.kuka.fr

India

KUKA India Pvt. Ltd.
Office Number-7, German Centre,
Level 12, Building No. - 9B
DLF Cyber City Phase III
122 002 Gurgaon
Haryana
India
Tel. +91 124 4635774
Fax +91 124 4635773
info@kuka.in
www.kuka.in

Italy

KUKA Roboter Italia S.p.A.
Via Pavia 9/a - int.6
10098 Rivoli (TO)
Italy
Tel. +39 011 959-5013
Fax +39 011 959-5141
kuka@kuka.it
www.kuka.it

Japan

KUKA Japan K.K.
YBP Technical Center
134 Godo-cho, Hodogaya-ku
Yokohama, Kanagawa
240 0005
Japan
Tel. +81 45 744 7531
Fax +81 45 744 7541
info@kuka.co.jp

Canada

KUKA Robotics Canada Ltd.
2865 Argentia Road, Unit 4-5
Mississauga
Ontario L5N 8G6
Canada
Tel. +1 905 858-5852
Fax +1 905 858-8581
KUKAFocusCenter@KUKARobotics.com
www.kukarobotics.ca

Korea

KUKA Robotics Korea Co. Ltd.
RIT Center 306, Gyeonggi Technopark
1271-11 Sa 3-dong, Sangnok-gu
Ansan City, Gyeonggi Do
426-901
Korea
Tel. +82 31 501-1451
Fax +82 31 501-1461
info@kukakorea.com

Malaysia

KUKA Robot Automation (M) Sdn Bhd
South East Asia Regional Office
No. 7, Jalan TPP 6/6
Taman Perindustrian Puchong
47100 Puchong
Selangor
Malaysia
Tel. +60 (03) 8063-1792
Fax +60 (03) 8060-7386
info@kuka.com.my

Mexico

KUKA de México S. de R.L. de C.V.
 Progreso #8
 Col. Centro Industrial Puente de Vigas
 Tlalnepantla de Baz
 54020 Estado de México
 Mexico
 Tel. +52 55 5203-8407
 Fax +52 55 5203-8148
 info@kuka.com.mx
 www.kuka-robotics.com/mexico

Norway

KUKA Sveiseanlegg + Roboter
 Sentrumsvegen 5
 2867 Hov
 Norway
 Tel. +47 61 18 91 30
 Fax +47 61 18 62 00
 info@kuka.no

Austria

KUKA CEE GmbH
 Gruberstraße 2-4
 4020 Linz
 Austria
 Tel. +43 732 784 752 0
 Fax +43 732 793 880
 KUKAAustriaOffice@kuka.com
 www.kuka.at

Poland

KUKA CEE GmbH Poland
 Spółka z ograniczoną odpowiedzialnością
 Oddział w Polsce
 Ul. Porcelanowa 10
 40-246 Katowice
 Poland
 Tel. +48 327 30 32 13 or -14
 Fax +48 327 30 32 26
 ServicePL@kuka-roboter.de

Portugal

KUKA Robots IBÉRICA, S.A.
 Rua do Alto da Guerra n° 50
 Armazém 04
 2910 011 Setúbal
 Portugal
 Tel. +351 265 729 780
 Fax +351 265 729 782
 info.portugal@kukapt.com
 www.kuka.com

Russia

KUKA Russia OOO
1-y Nagatinskiy pr-d, 2
117105 Moskau
Russia
Tel. +7 495 665-6241
support.robotics.ru@kuka.com

Sweden

KUKA Svetsanläggningar + Robotar AB
A. Odhners gata 15
421 30 Västra Frölunda
Sweden
Tel. +46 31 7266-200
Fax +46 31 7266-201
info@kuka.se

Switzerland

KUKA Roboter CEE GmbH
Linz, Zweigniederlassung Schweiz
Heinrich Wehrli-Strasse 27
5033 Buchs
Switzerland
Tel. +41 62 837 43 20
info@kuka-roboter.ch

Slovakia

KUKA CEE GmbH
organizačná zložka
Bojnická 3
831 04 Bratislava
Slovakia
Tel. +420 226 212 273
support.robotics.cz@kuka.com

Spain

KUKA Iberia, S.A.U.
Pol. Industrial
Torrent de la Pastera
Carrer del Bages s/n
08800 Vilanova i la Geltrú (Barcelona)
Spain
Tel. +34 93 8142-353
comercial@kukarob.es

South Africa

Jendamark Automation LTD (Agency)
76a York Road
North End
6000 Port Elizabeth
South Africa
Tel. +27 41 391 4700
Fax +27 41 373 3869
www.jendamark.co.za

Taiwan

KUKA Automation Taiwan Co. Ltd.
1F, No. 298 Yangguang ST.,
Nei Hu Dist., Taipei City, Taiwan 114
Taiwan
Tel. +886 2 8978 1188
Fax +886 2 8797 5118
info@kuka.com.tw

Thailand

KUKA (Thailand) Co. Ltd.
No 22/11-12 H-Cape Biz Sector Onnut
Sukhaphiban 2 road, Prawet
Bangkok 10250
Thailand
Tel. +66 (0) 90-940-8950
HelpdeskTH@kuka.com

Czech Republic

KUKA Roboter CEE GmbH
organizační složka
Pražská 239
25066 Zdiby
Czech Republic
Tel. +420 226 212 273
support.robotics.cz@kuka.com

Hungary

KUKA HUNGÁRIA Kft.
Fő út 140
2335 Taksony
Hungary
Tel. +36 24 501609
Fax +36 24 477031
info@kuka-robotics.hu

USA

KUKA Robotics Corporation
51870 Shelby Parkway
Shelby Township
48315-1787
Michigan
USA
Tel. +1 866 873-5852
Fax +1 866 329-5852
info@kukarobotics.com
www.kukarobotics.com

UK

KUKA Robotics UK Ltd
Great Western Street
Wednesbury West Midlands
WS10 7LL
UK
Tel. +44 121 505 9970
Fax +44 121 505 6589
service@kuka-robotics.co.uk
www.kuka-robotics.co.uk

Index

A

Appendix..... 104

C

CAST_FROM()..... 35, 47
 CAST_TO()..... 33, 47
 Client mode..... 12, 32
 Command reference..... 106
 Communication..... 9
 Configuration..... 20
 Configuration examples..... 43
 Configuration, Ethernet connection..... 9, 21
 Connection, monitoring..... 10

D

Data exchange..... 10
 Data exchange, functions..... 28
 Data stream..... 7
 Defragmentation..... 40
 Diagnosis..... 55
 Diagnostic data, displaying..... 55
 Diagnostic monitor (menu item)..... 55
 Documentation, industrial robot..... 6

E

EKI..... 7
 EKI_CHECK()..... 38, 41, 113
 EKI_CheckBuffer()..... 30, 114, 115
 EKI_Clear()..... 37, 106, 107, 113
 EKI_ClearBuffer()..... 37, 114
 EKI_Close()..... 32, 106
 EKI_GetBool()..... 109, 110
 EKI_GetBoolArray()..... 110
 EKI_GetFrame()..... 112
 EKI_GetFrameArray()..... 112, 113
 EKI_GetInt()..... 110
 EKI_GetIntArray()..... 110, 111
 EKI_GetReal()..... 111
 EKI_GetRealArray()..... 111
 EKI_GetString()..... 35, 112
 EKI_Init()..... 30
 EKI_Lock()..... 114
 EKI_Open()..... 32, 64, 66, 106
 EKI_Send()..... 33, 78–80, 83, 84, 107, 108
 EKI_SetBool()..... 108
 EKI_SetFrame()..... 108
 EKI_SetInt()..... 108, 109
 EKI_SetReal()..... 109
 EKI_SetString()..... 109
 EKI_STATUS..... 38
 EKI_Unlock()..... 114
 End string..... 7
 EOS..... 7

Error protocol..... 56
 Error response, programming..... 41
 Error treatment..... 13
 Ethernet..... 7
 Ethernet connection, configuration..... 9, 21
 Ethernet, interfaces..... 20
 EthernetKRL_Server.exe..... 43
 Event messages..... 13, 39
 Examples, integrating..... 43

F

Features..... 9
 FIFO..... 7, 11
 Fragmentation..... 40
 Functions..... 9
 Functions, data exchange..... 28

I

Installation..... 16
 Installation via smartHMI..... 17
 Installation via WorkVisual..... 16
 Intended use..... 13
 Introduction..... 6
 IP..... 8

K

KLI..... 7, 20
 Knowledge, required..... 6
 KR C..... 7
 KRL..... 7
 KUKA Customer Support..... 116
 KUKA Service..... 116

L

Licenses..... 8
 LIFO..... 7, 11, 40
 Logbook..... 56
 Lost connection..... 9

M

Memory, increasing..... 104
 Messages..... 56
 Messages, deactivating..... 105
 Monitoring, connection..... 10

N

Network connection..... 20

O

Open source..... 8
 Overview, KUKA.Ethernet KRL 3.1..... 9

P

Ping.....	10
Product description.....	9
Program examples.....	43
Programming.....	21
Programming tips.....	29
Protocol types.....	12

S

Safety.....	15
Safety instructions.....	6
Saving data.....	11
Server mode.....	12, 32
Server program.....	43
Server program, communication parameters	45
Server program, user interface.....	44
smartHMI.....	7
Socket.....	7
Support request.....	116
System requirements.....	16

T

Target group.....	6
TCP/IP.....	8
Terms used.....	7
Terms, used.....	7
Time stamp.....	39
Trademarks.....	8
Training.....	6

U

UDP/IP.....	8
Uninstalling via smartHMI.....	18
Uninstalling via WorkVisual.....	17
Updating via smartHMI.....	17
Updating via WorkVisual.....	16
Use, intended.....	13

W

Warnings.....	6
---------------	---

X

XML.....	8
XPath.....	8, 24, 27